Universidad Nacional Mayor de San Marcos

Universidad del Perú. Decana de América

Facultad de Ingeniería de Sistemas e Informática

Escuela Profesional de Ingeniería de Software

Un navegador de realidad aumentada para aplicaciones basadas en marcadores aplicando el estándar ARML 2.0

TESIS

Para optar el Título Profesional de Ingeniera de Software

AUTOR

Ángela Victoria CÓRDOVA GONZALES

ASESOR

Lenis Rossi WONG PORTILLO

Lima, Perú

2017

Referencia bibliográfica

Córdova, A. (2017). *Un navegador de realidad aumentada para aplicaciones basadas en marcadores aplicando el estándar ARML 2.0*. [Tesis de pregrado, Universidad Nacional Mayor de San Marcos, Facultad de Ingeniería de Sistemas e Informática, Escuela Profesional de Ingeniería de Software]. Repositorio institucional Cybertesis UNMSM.

1012

UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS
FACULTAD DE INGENIERÍA DE SISTEMAS E INFORMÁTICA
ESCUELA PROFESIONAL DE INGENIERÍA DE SOFTWARE

Acta de Sustentación de Tesis

170

Siendo las 5:55 del día 24 de febrero del año 2017, se reunieron los docentes designados como miembros de Jurado de la Tesis, presidido por el Mg. Jorge Díaz Muñante, Lic. Carlos Enrique Yañez Duran (Miembro), y la Mg. Lenis Rossi Wong Portillo (Miembro Asesor) para la sustentación de la Tesis intitulada: **"UN NAVEGADOR DE REALIDAD AUMENTADA PARA APLICACIONES BASADAS EN MARCADORES APLICANDO EL ESTÁNDAR ARML 2.0** por la Bach. **Ángela Victoria Córdova Gonzales** para optar el Título Profesional de Ingeniero de Software.

Acto seguido de la exposición de la Tesis, el Presidente invitó a la Bachiller a dar respuesta a las preguntas establecidas por los Miembros de Jurado.

La Bachiller en el curso de sus intervenciones demostró pleno dominio del tema, al responder con acierto y fluidez a las observaciones y preguntas formuladas por los señores miembros del Jurado.

Finalmente habiéndose efectuado la calificación correspondiente por los miembros de Jurado, la Bachiller obtuvo la nota de 18 (En letras) DIECIOCHO

A continuación el Presidente del Jurado, Mg. JORGE DÍAZ declara a la Bachiller **Ingeniero de Software.**

Siendo las 6:45 p.m. horas, se levantó la sesión.

Mg. Jorge Díaz Muñante
Presidente

Lic. Carlos Enrique Yañez Duran
Miembro

Mg. Lenis Rossi Wong Portillo
Miembro Asesor

Esta investigación está dedicada a mis padres María del Carmen y Aurelio a quienes amo y admiro profundamente.

AGRADECIMIENTOS

A la profesora Mg. Lenis Wong Portillo, por su apoyo en la orientación, revisión y dedicación para que el presente trabajo de investigación cumpla con los objetivos trazados.

A la profesora Dr. Nora La Serna por sus consejos y orientación durante mis años de estudiante y después de egresada.

A todos mis profesores quienes aportaron a mi desarrollo profesional, muchísimas gracias por todos los conocimientos brindados. Quiero hacer una mención especial a los profesores Mg. Erick Vicente de Tomas y al Mg. César Luza quienes me enseñaron durante los primeros años de carrera.

A mi familia, en especial a mis padres y a mi hermana Eleonora, por su apoyo constante durante la realización de esta investigación y por sus comentarios durante el desarrollo del trabajo.

A mis amigos y colegas quienes directa o indirectamente me ayudaron a culminar este trabajo.

Y finalmente A Dios y a la Virgen Auxiliadora.

Un navegador de realidad aumentada para aplicaciones basadas en marcadores aplicando el estándar ARML 2.0

RESUMEN

Los estudios muestran que uno de los mayores problemas del ecosistema de realidad aumentada es la falta de interoperabilidad entre los diferentes navegadores. Como respuesta a este problema, la organización *Open Geospatial Consortium* desarrolló el estándar ARML2.0.

A pesar de que este lenguaje fue liberado en el 2015, muy pocos navegadores y plataformas de realidad aumentada han realizado esfuerzos para implementar este estándar, tampoco se tienen noticias sobre futuras implementaciones.

Frente a este problema, el presente trabajo desarrolla un navegador de realidad aumentada para aplicaciones basadas en marcadores que interpreta escenas de realidad aumentada basadas en ARML 2.0.

La implementación se evalúa utilizando las pruebas definidas dentro del estándar ARML 2.0. Finalmente se realiza un caso de estudio donde puede verse la aplicación del navegador en un folleto aumentado.

Palabras claves: realidad aumentada, ARML2.0, marcadores de realidad aumentada.

An augmented reality browser for marker based augmented reality applications applying the ARML2.0 standard

ABSTRACT

Studies show that one of the greatest problems of the augmented reality ecosystem is the lack of interoperability between different browsers. In response to this problem, the Open Geospatial Consortium developed the standard ARML2.0.

Although this language was released in 2015, very few browsers and platforms of augmented reality have made efforts to implement this standard, neither have we had news about future developments.

This paper develops a marker based augmented reality browser that interprets augmented reality scenes based in ARML 2.0.

The implementation is evaluated using the tests defined in the ARML 2.0 standard. Finally a case study is shown, where we can see the application of the augmented reality browser in a brochure.

Key words: augmented reality, ARML2.0, augmented reality markers.

ÍNDICE GENERAL

LISTA DE FIGURAS

LISTA DE TABLAS

CAPÍTULO I
INTRODUCCIÓN

1.1 Antecedentes

La tecnología que permite la superposición de objetos virtuales sobre objetos del mundo real ha venido evolucionando desde 1968 cuando Ivan Shuterland propuso un sistema de realidad virtual que utilizaba un instrumento que se colocaba en la cabeza (*head-mounted display*) para visualizar objetos digitales [1]. Sin embargo, no fue sino hasta 1992, cuando Tom Caudell y David Mizell acuñan el término realidad aumentada, definiéndolo como aquello que permite que objetos generados por computador se superpongan sobre objetos del mundo real [2].

Los primeros sistemas basados en esta tecnología se enfocaron en buscar soluciones para un contexto específico. De esta manera encontramos sistemas especializados para los campos de la educación[3], medicina[4][5] e ingeniería[6][7][8], trabajos que se enfocaron en el desarrollo de técnicas para el reconocimiento de marcadores [9][10][11] o que exploraron la tecnología en diferentes plataformas, cómo la plataforma móvil [12].

A partir de 1999 se desarrollaron tecnologías y sistemas de realidad aumentada genéricos, es decir, que pueden ser aplicables a cualquier tipo de industria. Claro ejemplo de ello, son los *frameworks* para reconocimiento de marcadores como ARToolKit[13] y Alvar [14] y las diferentes propuestas de navegadores de realidad aumentada tales como [1] [15].

En la actualidad, el mercado hace uso de plataformas de realidad aumentada que tienen principalmente navegadores de realidad aumentada, que se integran con servicios en la nube, permitiendo la aplicación de esta tecnología en contextos de geolocalización y reconocimiento de imágenes. Los casos más importantes son el de Wikitude[16] y el de Layar [17]. También hay una tendencia creciente, en utilizar a las tecnologías de la web como la plataforma de facto para aplicaciones de realidad aumentada [18].

Sin embargo, a pesar de que estas plataformas han presentado un gran avance en el desarrollo de la tecnología, la industria de realidad aumentada aún presenta retos que dificultan su adopción masiva. Algunos de estos retos están relacionados a la experiencia de usuario y a la información que es necesaria para una aplicación de realidad aumentada [19]

Con relación a la experiencia de usuario, uno de los aspectos más resaltantes, es que en la actualidad no es posible "que el contenido de realidad aumentada de autores independientes pueda ser visto de manera conjunta" en diversas aplicaciones, sino que los desarrolladores tienen que optar por alguno de los dos modelos existentes, ya sea escogiendo plataformas específicas de realidad aumentada o integrando esta tecnología en aplicaciones web [19] [20].

Otro punto de dificultad ha sido "la creación, acceso, y administración" de la información que necesita una aplicación de realidad aumentada, ya que es necesario conocer no sólo la información sobre un objeto que se quiere aumentar, sino también la información sobre "la realidad que se está aumentando, y cómo esta se relaciona con la data interna de la aplicación"[19].

Frente a estos problemas, el *Open Geospatial Consortium*, creó en el 2015 el estándar ARML2.0, estándar que define "una codificación no propietaria, para que los proveedores de realidad aumentada puedan especificar la apariencia visual y los objetos del mundo real sobre los objetos virtuales en una escena de realidad aumentada" [21] y tiene aplicación en contextos de realidad aumentada basada en la geolocalización y en la visión. El estándar es un primer paso para "proveer un mecanismo que permita unir el contenido del mundo real con el contenido del mundo virtual".[19]

En la actualidad, el estándar ha sido implementado parcialmente en algunos navegadores de realidad aumentada, únicamente para contextos basados en la localización [22].

1.2 Definición del problema

El estándar ARML2.0 ha sido creado como una herramienta que ayuda a dar un primer paso en la solución de problemas de la industria de realidad aumentada tales como, la experiencia de usuario y la dificultad para la asociación de información entre el mundo real y el mundo virtual. Sin embargo, a pesar de que el estándar define mecanismos para su aplicación en distintos contextos, este sólo ha sido implementado parcialmente, mediante una prueba de concepto para aplicaciones basadas en la geo-localización [22][23], faltando una implementación y validación del mismo para aplicaciones basadas en la visión.

1.3 Objetivos

1.3.1 Objetivo general

Desarrollar un navegador de realidad aumentada para aplicaciones basadas en la visión, que reconozca marcadores mediante el estándar ARML 2.0.

1.3.2 Objetivos específicos

De igual manera, en base al objetivo general, se plantean los siguientes objetivos específicos:

- OE1: Revisar la literatura existente sobre los *frameworks*, navegadores y plataformas de realidad aumentada.
- OE2: Desarrollar un navegador de realidad aumentada para el sistema operativo Android a partir de la versión 5.0.
- OE3: Desarrollar un navegador de realidad que permita el reconocimiento de marcadores.
- OE4: Extender un *framework* opensource de realidad aumentada, para que permita el renderizado de escenas basadas en ARML 2.0.

1.4 Justificación

El desarrollo de aplicaciones de realidad aumentada basadas en el estándar ARML2.0 permite separar la descripción de las escenas de realidad aumentada de la interpretación de las mismas, esto facilita la experiencia de usuario, al lograr que un mismo software pueda reconocer contenido elaborado por terceros. Facilita además, la identificación y administración de los diferentes elementos involucrados en la escena y las relaciones existentes entre dichos elementos. Como se ha mencionado anteriormente, el estándar aún no ha sido validado en contextos de aplicaciones basadas en la visión.

Así mismo, es importante resaltar que el desarrollo de un navegador de realidad aumentada basado en la visión ofrece múltiples oportunidades de aplicación en campos tan diversos como la medicina, el turismo, la educación, el marketing o la ingeniería.

El presente trabajo desarrolla un navegador de realidad aumentada para aplicaciones basadas en la visión utilizando marcadores, leyendo una escena descrita bajo el estándar ARML2.0.

1.5 Alcance

El presente trabajo tiene el siguiente alcance:

- Se desarrollará un navegador de realidad aumentada para el sistema operativo Android a partir de la versión 5.0.
- El navegador de realidad aumentada podrá ser utilizado para aplicaciones basadas en la visión utilizando marcadores.
- El navegador podrá realizar el seguimiento y reconocimiento de sólo un (01) marcador de realidad aumentada por cada frame de cámara obtenido.
- Se implementará un subconjunto del estándar ARML2.0, enfocándose en los elementos básicos y obligatorios definidos en el estándar para el reconocimiento de marcadores. En la Tabla 1 se especifican los elementos y atributos seleccionados:

Elemento	Tipo	Descripción
ARElement	Interfaz	Elemento del cual derivan todos los especificados en ARML2.0 Atributo: *id*
Anchor	Interfaz	Describe un enlace del mundo real con el mundo físico. Atributo: *enabled*
ARAnchor	Interfaz	Describe un tipo de *Anchor* tales como marcadores o imágenes. Atributo: *assets*
VisualAsset	Interfaz	Representaciones visuales del contenido aumentado en la pantalla. Atributo: *enabled, zOrder, Orientation*
Label	Clase	Describe una vista en HTML. Atributo: *href, src, viewportWidth*
Text	Clase	Describe texto plano a ser mostrado en pantalla. Atributos: *src, style*
Image	Clase	Describe una imagen a ser mostrada en pantalla. Atributo: *href*
Tracker	Clase	Permite describir el framework que se usa para rastrear un conjunto de trackables asociados. Atributo: *uri*
TrackableConfig	Clase	Describe la configuración asociada a un trackable. Atributos: *tracker, src, order*
Trackable	Clase	Describe un objeto que va a ser rastreado. Atributo: *config*

Tabla 1.- Elementos de ARML2.0 dentro del alcance

- No se encuentra dentro del alcance la parte *script* del estándar ARML 2.0.

- No se encuentra dentro del alcance la creación o modificación de algoritmos de reconocimiento y seguimiento de imágenes y marcadores. Se usará una biblioteca como soporte a estos puntos.

- No se encuentra dentro del alcance el desarrollo de una herramienta para la creación visual de escenas de realidad aumentada bajo el estándar ARML 2.0.

- No se encuentra dentro del alcance el desarrollo de una plataforma de realidad aumentada.

1.6 Organización de la Tesis

El presente trabajo está organizado de la siguiente manera:

El capítulo 1 presenta los antecedentes, el problema, la justificación y los objetivos que se buscan alcanzar con el trabajo. También se define que elementos están incluidos dentro y fuera del alcance.

El capítulo 2 presenta el marco teórico; donde se muestran algunos conceptos relacionados a realidad aumentada y se presenta un modelo de referencia con el cual se pueden comparar distintos *frameworks*. Se hace una pequeña introducción al sistema operativo Android y se presenta el estándar ARML 2.0

En el capítulo 3 se realiza el estado del arte, se presentan diferentes navegadores y plataformas de realidad aumentada realizando un estudio de cada uno. En este capítulo también analizan las diferentes bibliotecas de realidad aumentada que permiten el reconocimiento de marcadores.

El capítulo 4 presenta la metodología de trabajo y se describe el desarrollo de la solución utilizando los artefactos definidos en la misma. También se describe la implementación del navegador de realidad aumentada y se muestran los resultados que permiten validar la implementación. Finalmente se realiza una descripción del navegador de realidad aumentada a nivel funcional.

El capítulo 5 presenta una caso de estudio, que muestra la aplicación del navegador de realidad aumentada para publicidad.

Finalmente en el capítulo 6 se dan a conocer las conclusiones de la investigación y los trabajos futuros.

CAPÍTULO II
MARCO TEÓRICO

El presente capítulo presenta los conceptos necesarios para el desarrollo del presente trabajo. Se comenzará hablando sobre realidad aumentada, definiendo el concepto, los modelos y arquitecturas existentes. Se continuará con la presentación del estándar ARML 2.0, presentando la definición de los principales elementos que lo componen.

Finalmente se hará una introducción al sistema operativo Android, por ser el sistema operativo sobre el cual se desarrolla el presente trabajo.

2.1 Realidad aumentada

En este subcapítulo se introducirá la definición de realidad aumentada, se explicarán algunos modelos que muestran la relación entre el mundo virtual y el mundo real y finalmente se dará una clasificación de los diferentes tipos de realidad aumentada existentes.

2.1.1 Definición

Para el presente trabajo, se utilizará la definición que provee Gartner, que define a la realidad aumentada como "el uso en tiempo real de información en forma de texto, gráficos, audio u otras formas virtuales integradas con el mundo real, permitiendo al usuario interactuar con este en lugar de separarlo de él" [24].

2.1.2 Modelos conceptuales de la relación realidad-virtualidad

Para entender mejor el concepto, se debe tener en cuenta, cuál es la relación entre los objetos que se encuentran en el mundo real y el mundo virtual. Para explicar dicha

relación, nos basaremos en el trabajo de Paul Milgram et al. que proponen un línea de "*continuidad realidad-virtualidad*" donde se describe una escala donde se observan ambientes totalmente reales, totalmente virtuales y ambientes mixtos, el modelo propuesto puede verse en la Figura 1 [25].

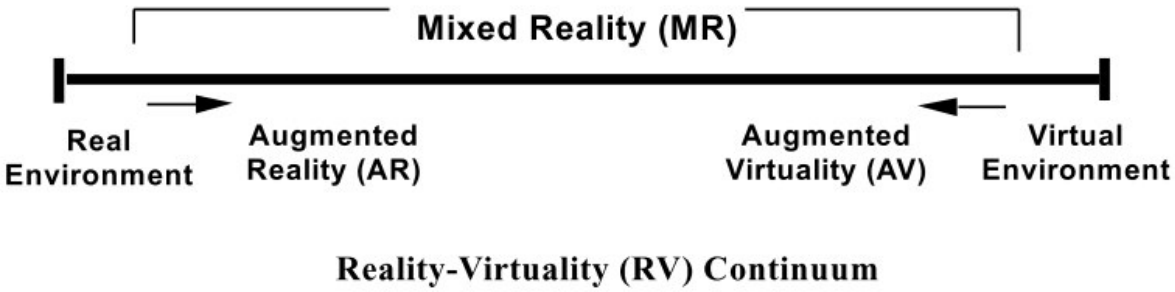


Figura 1 - Escala de continuidad realidad-virtualidad propuesta por Paul Milgram [25]

En la escala se definen dos ambientes diferentes:

- **Ambiente real (*Real environment*)** : En este tipo de ambientes se encuentran únicamente de objetos reales; se incluye además todo aquello que se puede observar al mirar el mundo real en persona, a través de una ventana o mediante algún tipo de video [25].

- **Ambiente virtual (*Virtual environment*):** En este tipo de ambientes se encuentran solamente de objetos virtuales, e incluye aquellos ambientes que son generados totalmente por computadora y simulaciones [25].

A medida que un ambiente se integra con el otro, comienzan a definirse diferentes tipos de realidades que se explican a continuación:

- **Realidad mixta (*Mixed reality*):** Representa un tipo de ambiente en el cual objetos del mundo virtual y del mundo real se combinan y se presentan juntos bajo una misma pantalla [25] y que aludiendo a la escala de continuidad realidad-virtualidad se encuentra en cualquier parte de ella excepto en los extremos.

- **Realidad aumentada (*Augmented reality*):** En el contexto de la escala de Milgram *et al* se observa que la realidad aumentada consiste en añadir elementos y objetos del ambiente virtual al ambiente real[25].

- **Virtualidad aumentada (*Augmented virtuality*):** Opuesto al concepto de realidad aumentada, en este caso se insertan objetos del mundo real sobre un ambiente totalmente virtual[25].

El modelo de Continuidad Realidad-Virtualidad no es el único en su tipo, existen otros modelos que permiten representar de forma gráfica las relaciones entre el ambiente virtual, el ambiente real y la forma en la que cada uno de estos ambientes pueden ser aumentados.

Vladimir Geroimenko propone un modelo donde explica cómo los objetos de cada ambiente influyen en el otro para producir lo que se conoce como Realidad Aumentada o Realidad Virtual (concepto equivalente al de Virtualidad Aumentada en el modelo de Milgram) Figura 2 [26].

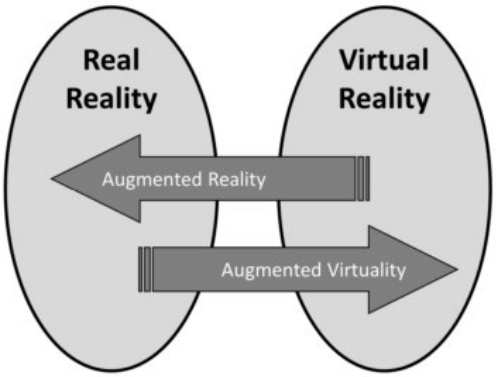


Figura 2 - Interrelación entre el ambiente real y el ambiente virtual [26]

2.1.3 Tipos de realidad aumentada

La clasificación de las diferentes aplicaciones de realidad aumentada, puede variar de autor en autor. A continuación se muestran algunas clasificaciones:

2.1.3.1 Clasificación basada en el tipo de aplicaciones

Esta clasificación es la que más comúnmente se encuentra en la literatura. Se definen 3 tipos de aplicaciones: aplicaciones basadas en la localización, basadas en el reconocimiento de imágenes, basados en la visión [27]. Procedemos a explicar cada una de ellas:

Aplicaciones de realidad aumentada basados en la Localización (*Location based*)

Este tipo de aplicaciones utiliza sensores de un dispositivo tales como la brújula, acelerómetro, GPS, para obtener la ubicación y posición de un usuario [27], a partir de estos datos se muestra información relevante a dicha ubicación.

Aplicaciones de realidad aumentada basada en imágenes (*Image based*)

Este tipo de aplicaciones utiliza el reconocimiento de imágenes para proveer información adicional sobre el objeto visualizado, esta información es después integrada con el mundo real para ofrecer una experiencia más rica. Una característica importante de este tipo de aplicaciones es que el contenido aumentado está desacoplado del campo visual observado, es decir, no se extrae información del campo visual (la imagen en sí) para obtener la información adicional (aumentada), sino que se compara el campo visual con alguna base de datos imágenes o similar y se obtiene la información buscada de la misma base de datos[27].

Aplicaciones de realidad aumentada basadas en la visión (*Vision based*)

Este tipo de aplicaciones utiliza algoritmos de visión computacional para reconocer la imagen capturada, una vez reconocida se procede a mostrar el objeto que ha sido relacionado con dicha imagen [27].

2.1.3.2 Clasificación basada en los sentidos

Este tipo de clasificación se basa en las percepciones humanas. En general, se define un tipo de aplicación por cada sentido como se aprecia en la Figura 3 [26]:

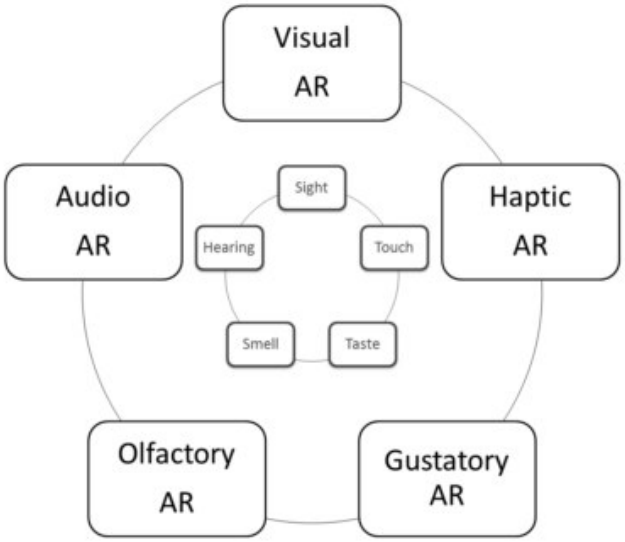


Figura 3 - Clasificación de realidad aumentada basada en los sentidos [26]

Realidad aumentada visual: Es el tipo más común de realidad aumentada. En este caso, gráficos generados por computador se insertan en el mundo real. Dichos gráficos pueden ser 2D o 3D [26].

Realidad aumentada auditiva: Busca embeber audio digital en el ambiente real [26].

Realidad aumentada táctil: Permite a los usuarios tocar y sentir objetos colocados en el ambiente real, para ello se necesita de algún dispositivo especial como guantes de realidad virtual [26].

Realidad aumentada olfativa: Aunque este tipo de realidad aumentada aún no se ha creado, en teoría permitiría acceder al olor del objeto real. Por ejemplo si se escanea una fotografía de una comida, se podría sentir el olor de la misma [26].

Realidad aumentada gustativa: Al igual que el anterior este tipo de realidad aumentada no se ha creado aún, pero permitiría a los usuarios saborear diferentes objetos reales [26].

2.1.4 Marcadores de realidad aumentada

En las aplicaciones de realidad aumentada basadas en la visión, se conoce como marcador a un elemento que permite rastrear objetos en el mundo real. Por lo general

estos elementos son cuadrados con patrones en blanco y negro que al ser reconocidos por un software de realidad aumentada muestran un objeto digital. El software de realidad aumentada es capaz de reconocer la ubicación del marcador en el mundo real y ubicar el objeto digital de acuerdo a la posición obtenida, de manera tal que se mantiene adherido al marcador incluso si éste cambia de posición [28][26]. La Figura 4 muestra ejemplos de marcadores del framework ARToolKit [29].

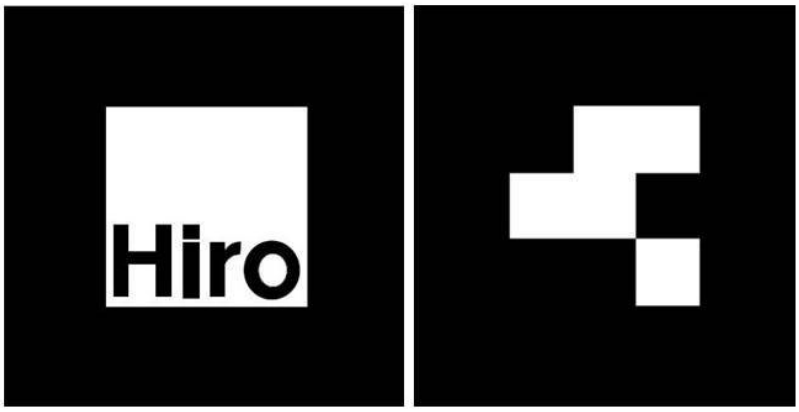


Figura 4 - Marcadores del framework ARToolKit [29]

El concepto de marcador también se utiliza para diferenciar a las aplicaciones que utilizan marcadores para el reconocimiento, de aquellas que no los utilizan, término conocido como *makerless augmented reality*; en este tipo de aplicaciones, no se utilizan los patrones en blanco y negro si no que se utilizan imágenes, texto e incluso rostros que al ser reconocidos permiten al software de realidad aumentada mostrar un objeto digital [26], un ejemplo puede verse en la Figura 5 donde se utiliza una imagen que permite acceder a un video y un botón de compra. [17].

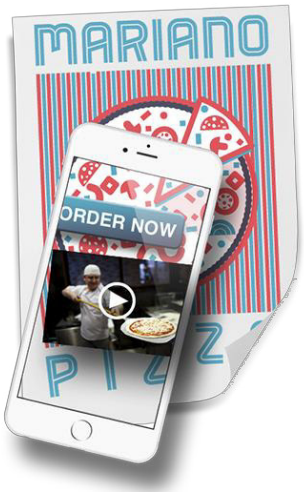


Figura 5 - Markerless Augmented Reality [17]

2.1.5 Modelo de referencia para sistemas de realidad aumentada

T. Reicher estableció un modelo de referencia donde se identifican 6 componentes básicos que se encuentran en todos los sistemas de realidad aumentada. Estos componentes siguen la estructura del patrón Modelo - Vista - Controlador, pero extiende este patrón con otros componentes que tienen responsabilidades propias en las aplicaciones basadas en esta tecnología [30]. Los subsistemas identificados son:

Subsistema de Aplicación: Este subsistema contiene toda la funcionalidad propia de la aplicación desarrollada. Es equivalente al subsistema modelo en el patrón MVC [30].

Subsistema de Interacción: Este subsistema es el encargado de procesar toda la información sobre la interacción que el usuario hace con el sistema. Es equivalente al subsistema controlador en el patrón MVC [30].

Subsistema de Presentación: Este subsistema es el encargado de presentar los objetos aumentados al usuario. Es equivalente con el subsistema vista del patrón MVC [30].

Subsistema de Seguimiento (*Tracking*): Este subsistema rastrea la posición del usuario y se encarga de comunicarla a los demás subsistemas [30].

Subsistema de Contexto: Este subsistema obtiene información sobre el contexto en el que se encuentra la aplicación y la comunica a los demás subsistemas [30]. Como ejemplos se puede tener el registro de la temperatura actual o el idioma del registrado en el dispositivo del usuario.

Subsistema Modelo del Mundo (*World model*): Este subsistema contiene información modelada sobre el mundo real. En las aplicaciones de realidad aumenta esta información describe una escena donde se identifican los objetos reales, los objetos virtuales y los diferentes atributos y relaciones de cada uno [30].

La Figura 6 muestra la equivalencia entre el patrón MVC y el modelo propuesto por Reicher [30].

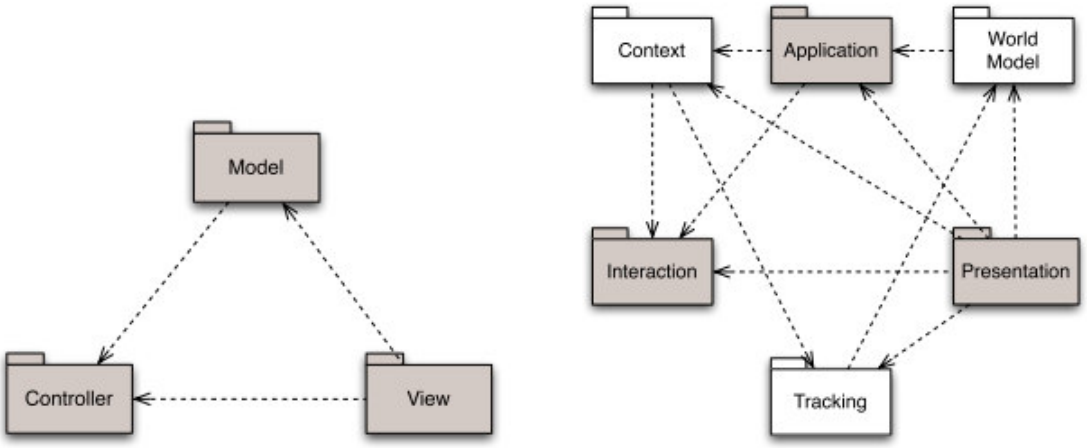


Figura 6 - Componentes de una aplicación de realidad aumentada según T. Reicher [30]

2.1.6 Arquitecturas de sistemas de realidad aumentada

Tomando como referencia la arquitectura genérica definida por Reicher, B. Butchart definió 4 tipos de arquitectura para sistemas de realidad aumentada [31], que se describen a continuación:

- **Arquitectura Gateway y Plataforma**

  En la arquitectura de tipo Gateway (Figura 7), la mayoría de los subsistemas de realidad aumentada se encuentran dentro de una aplicación cliente (un navegador de realidad aumentada). Por lo general esta aplicación tiene los subsistemas *Aplicación, Interacción, Tracking y Presentación*. Los subsistemas *Contexto* y *Mundo Modelo* son accedidos mediante APIs.

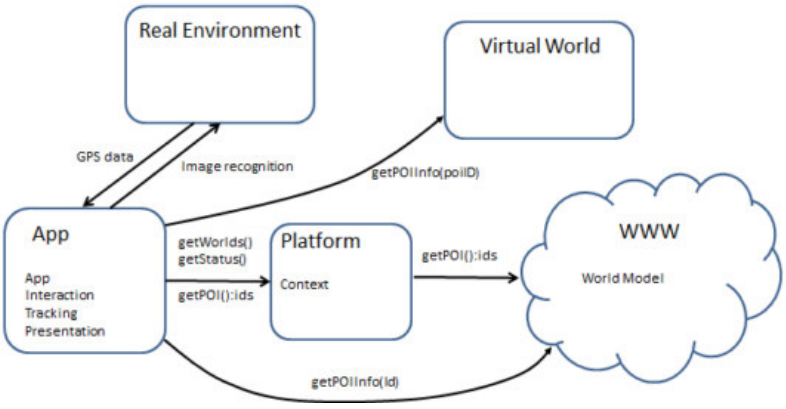


Figura 7.- Arquitectura Gateway [31]

Cuando el mundo modelo se convierte en una plataforma, este tipo de arquitectura cambia su nombre a Plataforma.

En ambas arquitecturas, el acceso al contenido aumentado, se da por medio de canales o mediante un sistema de reconocimiento de imágenes.

- **Arquitectura Web**

  Este tipo de arquitectura se da cuando se elimina el componente de plataforma, similar a lo que ocurre con los navegadores web. En estos casos no existe un ente que brinde un servicio de realidad aumentada. Los desarrolladores son los responsables de crear su contenido, hacerlo público y controlar el acceso al *Mundo Modelo*. La Figura 8 muestra un diagrama de este tipo de arquitectura.

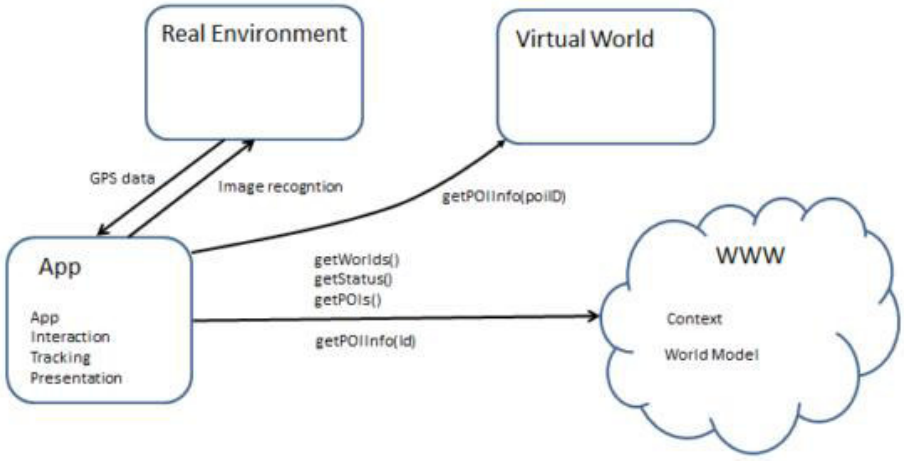


Figura 8 - Arquitectura Web [31]

- **Arquitectura Standalone**

  Este tipo de arquitectura incluye todos los sistemas de realidad aumentada de la arquitectura de referencia dentro de una aplicación cliente. Por lo general este tipo de sistemas está diseñado para realizar una funcionalidad específica. La Figura 9 muestra un diagrama con los componentes de esta arquitectura.

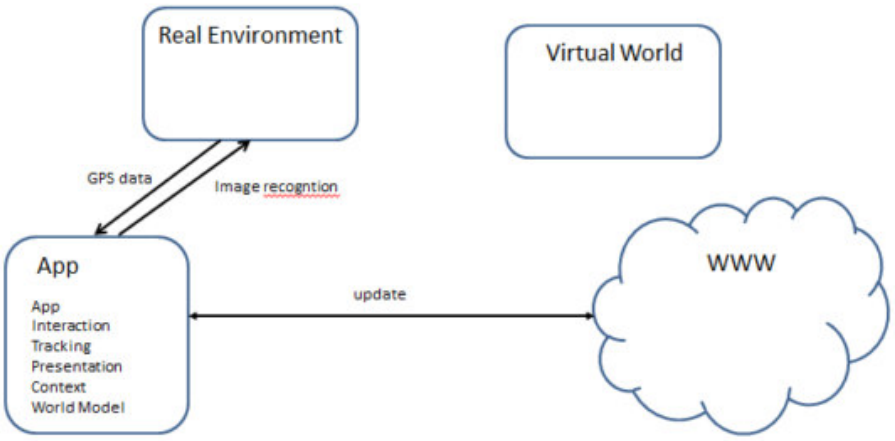


Figura 9 - Arquitectura Standalone [31]

2.2 Estándar ARML 2.0

2.2.1 Conceptos básicos

ARML 2.0 es un lenguaje estándar definido por la Open Geospatial Consortium que permite describir escenas de realidad aumentada basadas en la geolocalización y en la visión usando XML. Adicionalmente define parámetros basados en ECMAScript para la modificación de dichas escenas [32].

Este lenguaje define algunos conceptos para la descripción de escenas de realidad aumentada que se describen a continuación:

- ***Feature***: El estándar define a este elemento como "una abstracción de un fenómeno del mundo real" que contiene uno o más *Anchors* relacionados [32].

  Si hablamos de aplicaciones basadas en la geolocalización un *Feature* sería el lugar físico a reconocer, por ejemplo un museo.

- ***Anchor****:* Se refiere a "el registro de un *Feature* en el mundo real o en la pantalla" [32]. Existen dos tipos de *Anchors: ARAnchor y ScreenAnchor.*

  Un ***ARAnchor*** es una entidad a ser reconocida. Es de tipo ***Geometry*** cuando puede ser descrito con tuplas de coordenadas (por ejemplo de latitud y longitud para aplicaciones basadas en geolocalización), de tipo ***Trackable*** cuando puede

ser rastreado o reconocido (por ejemplo imágenes o marcadores para aplicaciones basadas en la visión) o de tipo ***RelativeTo*** cuando se requiere definir la posición de un objeto con respecto a otro.

Un ***ScreenAnchor*** es un elemento que siempre se mostrará en la pantalla del usuario independiente de su posición. Fue pensado para aquellas aplicaciones de realidad aumentad donde se debe mostrar información constantemente al usuario, por ejemplo el puntaje de un juego en pantalla.

- ***VisualAsset:*** Son "las representaciones de los *Features* (y los *Anchors* relacionados a estos) en la pantalla" [32] , es decir los elementos aumentados generados por computador.

  Están definidos 2 tipos de ***VisualAssets***: aquellos que se encuentran en 2 dimensiones se llaman ***VissualAsset2D*** y aquellos que se encuentran en 3 dimensiones se les denomina ***Model.***
  Dentro de los ***VissualAsset2D*** están definidos 4 tipos:

  - ***Label:*** Permite el modelado de contenido mediante HTML.
  - ***Fill:*** Representa un área geométrica de color.
  - ***Text:*** Texto plano.
  - ***Image:*** Una imagen.

El estándar nos presenta algunos ejemplos donde se puede ver la relación entre estos elementos, la Tabla 2 presenta de manera visual la relación para aplicaciones basadas en la geolocalización. La Tabla 3 muestra la relación para aplicaciones basadas en la visión [32].

Aplicaciones basadas en la geolocalización		
Feature		Para este caso, un *Feature* es el lugar real que quiere ser reconocido.
Anchor		El *Anchor* son las coordenadas de latitud y longitud del lugar (el *Feature*)
VisualAsset		Es el objeto generado por computador que quiere mostrarse.
Resultado		Cuando una aplicación de realidad aumentada basada en la geolocalización reconoce el *Anchor*, se muestra el objeto aumentado (*VisualAsset*)

Tabla 2 - Conceptos de ARML 2.0 aplicados a la geolocalización [32]

Aplicaciones basadas en la visión		
Feature	Los elementos de seguridad de un billete	Descripción del objeto a reconocer.
Anchor		*Anchor* es la representación física del objeto.
VisualAsset		*VisualAsset* es el objeto generado por computador que se mostrará
Resultado		Al reconocer los elementos de seguridad del billete definidos, se muestran los objetos aumentados.

Tabla 3 - Conceptos de ARML 2.0 aplicados a la visión [32]

Para el caso de aplicaciones de realidad aumentada basadas en la visión, el estándar también define un elemento llamado ***Trackable***, que permite describir el framework utilizado para poder reconocer y rastrear la posición de objetos definidos en el elemento ***Anchor*** de tipo ***Tracker***.

Todos los elementos anteriormente descritos se encuentran representados en un modelo basado en objetos. Este modelo define un diagrama de clases donde se encuentran todas las relaciones entre los diferentes elementos del estándar y las propiedades definidas en

cada uno. Este modelo permite entender de manera gráfica las relaciones expuestas dentro de un archivo XML basado en ARML 2.0, puede verse en la Figura 10.

La descripción de todos los elementos que componen el estándar ARML2.0 puede encontrarse en el ANEXO I.

2.2.2 Pruebas de conformidad de ARML 2.0

ARML 2.0 define tres conjuntos de pruebas sobre las cuales se puede validar la implementación de un software basado en este estándar. A continuación se detalla un resumen de cada una de ellas:

- **Pruebas de conformidad sobre la codificación:** Este tipo de pruebas son utilizadas para validar que la codificación de un archivo XML cumple con lo definido en el estándar ARML 2.0.

- **Pruebas de conformidad sobre la implementación descriptiva:** Este tipo de pruebas permite validar la correcta implementación de un software capaz de leer archivos de ARML 2.0 que solamente tengan formato descriptivo. Este es el tipo de pruebas que se utilizarán para el presente trabajo.

- **Pruebas de conformidad sobre la implementación descriptiva y con scripts:** Este tipo de pruebas permite validar la correcta implementación de un software capaz de leer archivos de ARML 2.0 que tienen una parte descriptiva y otra basada en scripts.

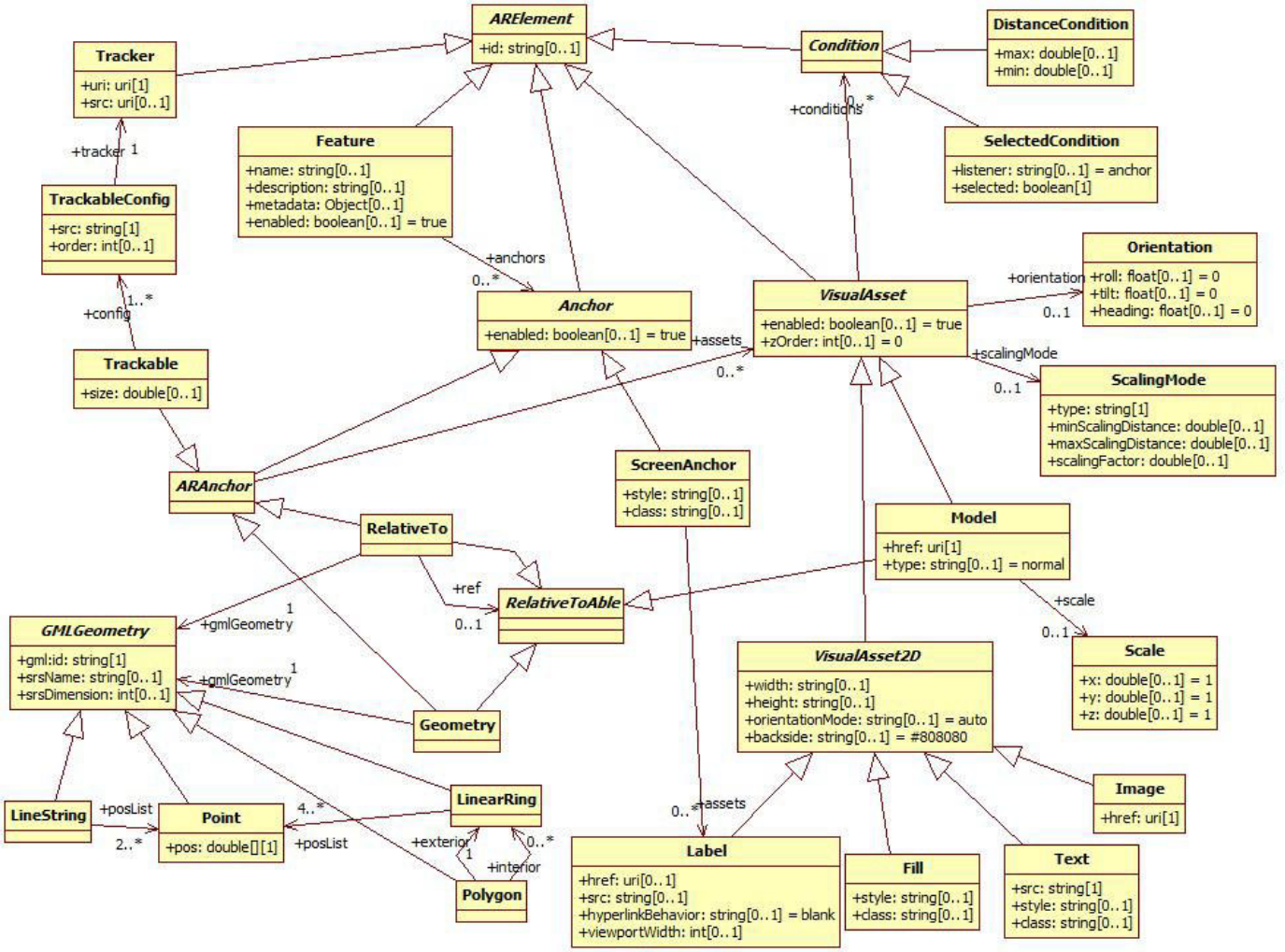


Figura 10.- Diagrama de clases del estándar ARML 2.0 [32]

2.2.3 ARML 2.0 para aplicaciones basadas en la visión

Un ejemplo de la descripción de una escena de realidad aumentada usando el estándar puede verse en la Figura 11, donde se aumenta un modelo 3D llamado *myModel.dae* sobre un marcador llamado *myMarker.jpg*. La descripción de los elementos en esta escena puede verse en la Tabla 4.

```
<arml>
  <ARElements>
    <Tracker id="defaultImageTracker">
      <uri
        xlink:href="http://www.opengis.net/arml/tracker/genericImageTracker" />
    </Tracker>
    <Trackable>
      <assets>
        <Model>
          <href xlink:href="http://www.myserver.com/myModel.dae" />
        </Model>
      </assets>
      <config>
        <tracker xlink:href="#defaultImageTracker" />
        <src>http://www.myserver.com/myMarker.jpg</src>
      </config>
      <size>0.20</size>
    </Trackable>
  </ARElements>
</arml>
```

Figura 11 Escena de realidad aumentada usando ARML 2.0 [32]

Campo	Descripción
ARML	Etiqueta obligatoria que indica que se está utilizando ARML 2.0
ARElements	Etiqueta de tipo interface obligatoria en ARML2.0
Tracker	Referencia a la funcionalidad que va a ser utilizada por el framework para reconocer el objeto virtual
Uri	URI que da acceso a la implementación del Tracker utilizado por el sistema de realidad aumentada.

Trackable	Identifica al objeto del mundo real a ser reconocido.
Assets	Identifica los objetos virtuales relacionados
Model	Identifica a un objeto 3D
Href	Hipervínculo donde se encuentra el objeto 3D
Config	Parámetros de configuración del elemento Trackable
Tracker	Relaciona el elemento Trackable con un elemento Trackable.
Src	Identifica al objeto del mundo real que debe ser reconocido.
Size	La medida del elemento Trackable en metros.

Tabla 4.- Descripción de los elementos de una escena de realidad aumentada usando ARML2.0

2.2.4 Relación de ARML 2.0 con otros lenguajes de realidad aumentada

ARML 2.0 no es el primer lenguaje que permite describir escenas de realidad aumentada. Trabajos como [33][15][18] han propuesto otras alternativas para este propósito, creando sus propios lenguajes o utilizando tecnologías ya existentes como HTML. Sin embargo, ARML 2.0 provee características únicas tales como la capacidad de estar orientado a aplicaciones de realidad aumentada basadas en la visión y en la localización, provee un conjunto de pruebas de validación para cada componente del lenguaje y considera la integración con otros estándares tales como GML, HTML y Collada.

Una descripción y un análisis más extenso de los otros lenguajes y las diferencias con el estándar ARML2.0 puede encontrarse en el ANEXO II.

2.3 Sistema operativo Android

En este subcapítulo se presentarán conceptos relacionados al sistema operativo Android. Se comenzará hablando sobre su historia, su arquitectura y se mencionarán algunos de los componentes básicos para el desarrollo de aplicaciones en esta plataforma.

2.3.1 Definición

Android es una de las plataformas móviles más populares del mundo [34]. Collins et al. lo definen como "un conjunto completo de software para dispositivos móviles: un sistema operativo, middleware, y aplicaciones básicas para móviles" [35]. Está basado en la versión del kernel de Linux 2.6 y provee servicios fundamentales como seguridad, administración de memoria, administración de procesos, servicios de red entre otros. Utiliza la máquina virtual Dalvik que ejecuta las aplicaciones Android que son escritas en Java[36]. Además, provee herramientas de desarrollo y un SDK que son partes importantes de la plataforma. Android no está diseñado para un dispositivo en particular, si no que puede ser adaptado a diferentes configuraciones de hardware (móviles, tablets, netbooks, entre otros) [35].

2.3.2 Componentes de una aplicación Android

Android provee un marco de trabajo de desarrollo, definiendo componentes básicos para el desarrollo de aplicaciones, estos componentes son: Actividades, Intents, Vistas, Servicios, Proveedores de contenido, Receptores de difusión y el Manifiesto. A continuación se explica cada uno de ellos:

Actividad: Es una única interfaz de una aplicación. Generalmente es esta interfaz ocupa la totalidad de la pantalla del dispositivo (aunque puede haber pantallas flotantes) y su objetivo es recibir y administrar los eventos de interfaz de usuario [37].

Intent: Es un objeto que encapsula un mensaje. Su objetivo principal es iniciar o proveer comunicación entre otras aplicaciones y/o actividades. [37]

Vista: Son elementos de interfaz de usuario que dan forma al control y la apariencia de la aplicación. Un ejemplo puede ser un botón o un campo de texto [35].

Servicio: Es un elemento que se ejecuta en segundo plano, pero que no tiene asociada ninguna interfaz de usuario. Un ejemplo puede ser la reproducción de música (no es necesario tener la interfaz del reproductor en primer plano para escuchar la música) o el monitoreo de red [37].

Proveedor de contenido: Es una capa de abstracción que permite exponer los datos de una aplicación para que sean usados por otras aplicaciones [37].

Receptor de difusión: Es un componente que recibe y administra adecuadamente todos los intents del sistema que no son enviados directamente a una aplicación o actividad [37].

Manifiesto: Es un archivo XML que todas las aplicaciones deben definir y provee opciones de configuración para la aplicación [35].

2.3.3 Ciclo de vida de las actividades de Android

Las actividades representan uno de los componentes más importantes de las aplicaciones Android. Tal y como se mencionó anteriormente, una actividad es una interfaz de usuario que provee método para controlar todas los eventos e interacciones que ocurren sobre ella. Así mismo tiene una serie de métodos que definen las acciones o pasos a desarrollar cuando se quiere crear una nueva actividad o cuando se quiere eliminar una; esta serie de métodos se pueden explicar mejor en el Ciclo de Vida de las Actividades Android (ver Figura 12)

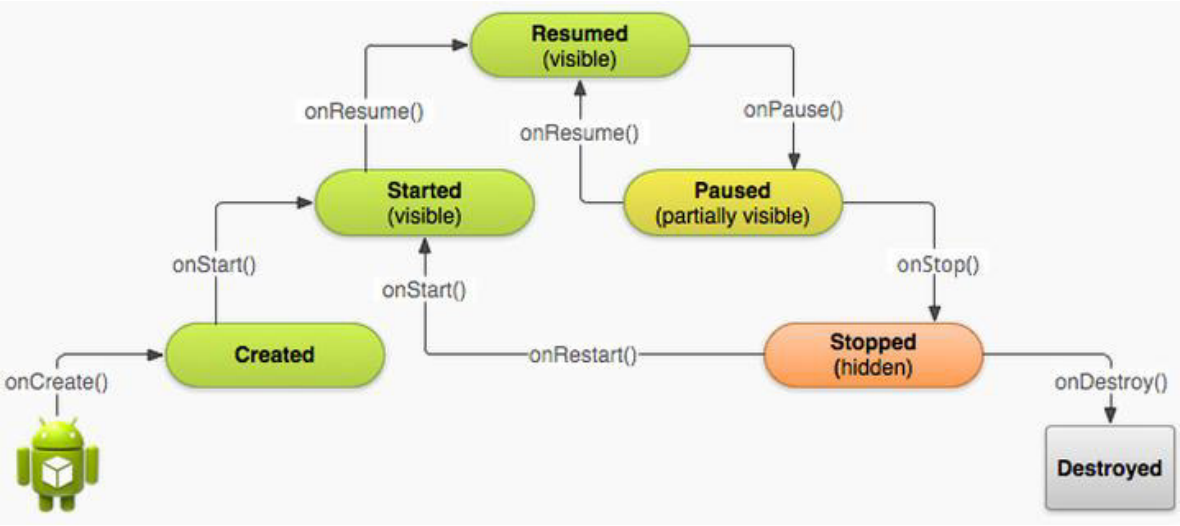


Figura 12 - Ciclo de vida de una actividad [38]

El ciclo de vida define 6 estados: Creado, Iniciado, Reanudado, Pausado, Parado y Destruido. También se definen una serie de métodos que dependiendo del estado donde se encuentre la actividad, harán que esta pase a un estado diferente. El comportamiento de estos métodos debe ser sobrescrito por los desarrolladores de aplicaciones [39].

Estado Creado (*Created*): Es un estado transitorio. Una actividad llega a este después de que el usuario a seleccionado una aplicación que quiere iniciar; en ese momento se ejecuta el método *onCreate*. La actividad todavía no es visible.

Estado Iniciado (*Started*): Estado transitorio. Una vez que se termina de ejecutar el método *onCreate* se ejecuta el método *onStart* que hace que la aplicación se coloque en estado Iniciado. La actividad ya se encuentra visible.

Estado Reanudado (*Resumed*): Se llega a este estado mediante el método *onResumed* que se ejecuta después del método *onStart*. En este estado la actividad se encuentra visible y el usuario ya puede interactuar con ella [40].

Estado Pausado (*Paused*): Se llega a este estado de manera inmediata cuando el usuario inicia otra actividad que se superpone total o parcialmente sobre la actividad actual en uso, también cuando la nueva actividad iniciada es semitransparente. Cuando alguno de estas dos situaciones sucede, el sistema ejecuta inmediatamente el método

onPause, en este método se deberán ejecutar algunas tareas de persistencia de datos o se deberá parar algunas actividades como la reproducción de video [38]. Generalmente este el primer indicio de que el usuario dejará la actividad.

Estado Parado (*Stopped*): El sistema ejecuta el método *onStop* cuando la actividad está totalmente oculta. Esto puede suceder porque el usuario abrió otra aplicación, el usuario inició otra actividad que oculta totalmente la predecesora o cuando el usuario recibe una llamada; todos estos indicios hacen que la actividad llegue al estado de Parado. Generalmente el método *onStop* deberá realizar la mayoría de liberación de recursos como la escritura en base de datos. Es importante saber que una actividad pausada puede ser automáticamente destruida por el sistema si es que este necesita recuperar memoria [41].

Estado Destruido (*Destroyed*): Se llega a este estado cuando el sistema ejecuta el método *onDestroy*. La actividad es completamente removida de la memoria del sistema.

2.4 Glosario de términos

- **Android:** Conjunto completo de software para dispositivos móviles, que incluye: un sistema operativo, middleware, y aplicaciones básicas para móviles

- **Aplicación móvil:** Programa, aplicación informática diseñada para un dispositivo móvil.

- **Framework:** Marco de trabajo que incluye herramientas y módulos de software para desarrollar algún programa específico.

- **Marcador:** Patrón que es reconocido por un software de realidad aumentada para obtener información relacionada. Hay dos tipos: técnicos y naturales.

- **Realidad aumentada:** Consiste en la superposición de objetos tales como gráficos generados por computadora, audio, video, enlaces web y cualquier otro tipo de información virtual (digital) sobre objetos del mundo real. Se accede a esta información mediante algún dispositivo electrónico.

- **Sistema operativo:** Programa o conjunto de programas que ofrece una serie de servicios para interactuar con algún dispositivo hardware y administrar las aplicaciones corriendo sobre él.

- **Software Development Kit - SDK:** Kit para desarrollo de software. Un conjunto de programas que permiten, facilitan el desarrollo de software.

CAPÍTULO III
ESTADO DEL ARTE

En este capítulo se analizarán todos los trabajos relacionados con los navegadores y sistemas de realidad aumentada existentes. También se estudian las bibliotecas que permiten realizar el registro y seguimiento de marcadores.

3.1 Navegadores y sistemas de realidad aumentada

En este subcapítulo se presentan las arquitecturas de algunos navegadores de realidad aumentada para dispositivos móviles y se describen de manera general los sistemas dentro de los cuales fueron desarrollados.

3.1.1 MARW

En [42] Karhu et al. proponen que las aplicaciones de realidad aumentada se basen en tecnologías de la web tales como HTML5, WebRTC, XML3D y JavaScript, mediante el uso de un navegador web genérico incorporado en el dispositivo móvil. Este enfoque permitiría abarcar un gran número de plataformas móviles bajo un mismo desarrollo.

En su trabajo desarrollan el framework MARW que permite la creación de aplicaciones basadas en la geolocalización y en la visión (para este último caso se hace uso del framework ALVAR). Para ello, se basaron en una arquitectura basada en capas y crearon una interfaz que permite a los desarrolladores acceder a diferentes servicios. La disposición de los elementos de esta arquitectura es descrita por los autores en la Figura 13 [42].

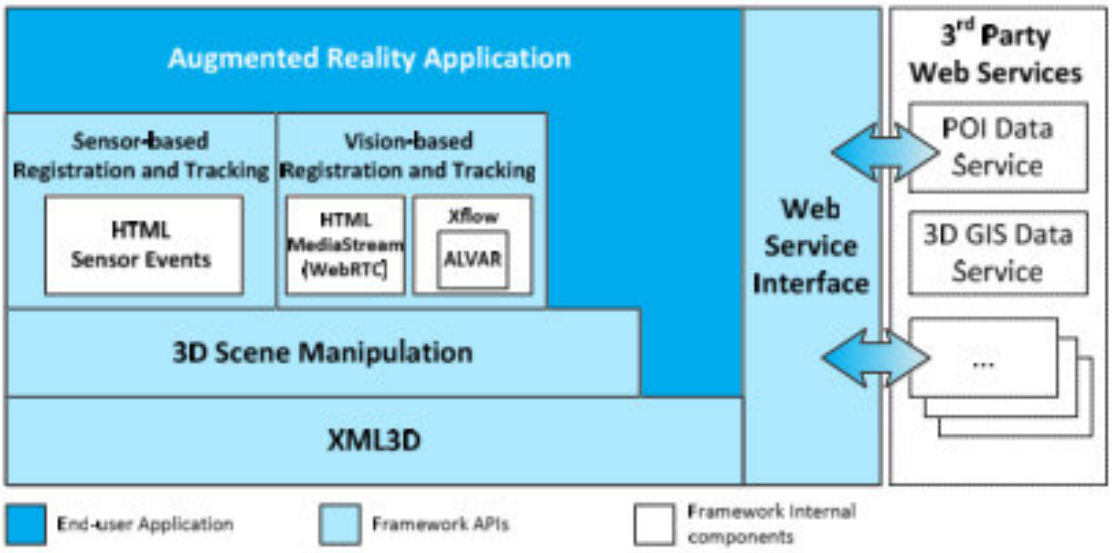


Figura 13 - Arquitectura del framework MARW [42]

El framework tiene los siguientes componentes:

- **Módulo para el registro de información de sensores:** Este módulo está dirigido a las aplicaciones de realidad aumentada basadas en la geolocalización. Utiliza HTML5 y las APIs de geolocalización y orientación definidas por la W3C [42].

- **Módulo para registro de información visual:** Este módulo está orientado a las aplicaciones de realidad aumentada basadas en la visión. Permite acceder a dispositivos para capturar información externa, obtiene información sobre puntos de detección de marcadores o imágenes existentes en el mundo real y finalmente procesa la data obtenida [42].

- **Módulo para la manipulación de escenas 3D:** Permite transformar la data obtenida desde los sensores o desde la cámara a formato XML3D [42].

- **Módulo de XML3D:** Logra la representación de los objetos aumentados en el lenguaje XML3D, este lenguaje es una extensión de HTML5.

- **Interfaz de servicios web:** Permite la comunicación de la aplicación con servicios web de terceros.

Los autores concluyen que el uso de un navegador web como un navegador de realidad aumentada permite un fácil desarrollo para múltiples plataformas, al basarse en tecnologías web. Sin embargo, reconocen que el uso de estas tecnologías puede introducir "ciertos problemas de rendimiento" en comparación a las aplicaciones desarrolladas nativamente [42].

3.1.2 ARGON

En [43] McIntyre et al. presentan el navegador para realidad aumentada ARGON para iOS. Este navegador se basa en el ecosistema de la web, y utiliza el lenguaje KARML para describir escenarios de realidad aumentada y la arquitectura KHARMA.

El navegador Argon sigue una arquitectura híbrida, donde algunas de sus características se encuentran desarrolladas nativamente, sin embargo, se exponen interfaces estos desarrollos mediante un intérprete JavaScript. El navegador usa una implementación de WebKit para el renderizado de los objetos aumentados mediante HTML y CSS3. Su arquitectura puede verse en la Figura 14 [43].

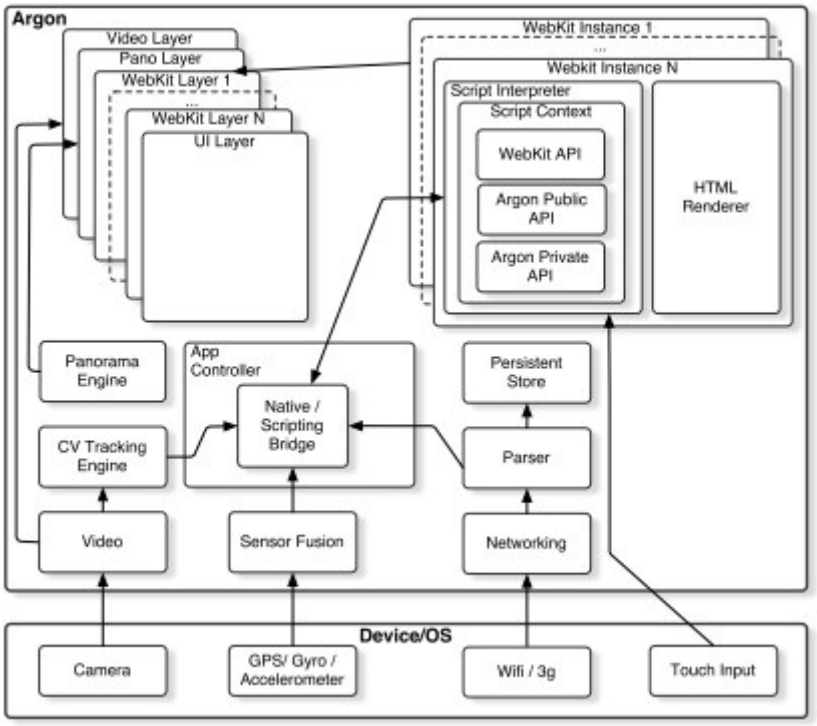


Figura 14 - Arquitectura de ARGON web browser [43]

Para el caso de aplicaciones basadas en la geolocalización, la arquitectura de Argon contempla la instanciación de múltiples instancias del navegador WebKit de manera que la información de múltiples fuentes independientes pueda ser desplegada en una misma vista.

Así mismo el navegador utiliza APIs JavaScript públicas y privadas que permiten la conexión entre el código nativo y las interfaces para la web.

3.1.3 Layar

Layar es una empresa que se especializa en la creación de aplicaciones realidad aumentada basadas en la geolocalización y aplicaciones basadas en la visión [44].

Layar ofrece una plataforma y un SDK para el desarrollo de aplicaciones de realidad aumentada para dicha plataforma.

En la Figura 15 se muestra la arquitectura de la plataforma Layar, que está separada en 5 capas: un navegador de realidad aumentada, una capa que contiene el servidor Layar, un sitio web de publicación de contenidos, una capa de proveedores de servicio y una capa de recursos. También ofrece 2 tipos de APIs que están expuestas a terceros: API de clientes y el API de desarrolladores [45]. A continuación se explica cada uno de estos componentes:

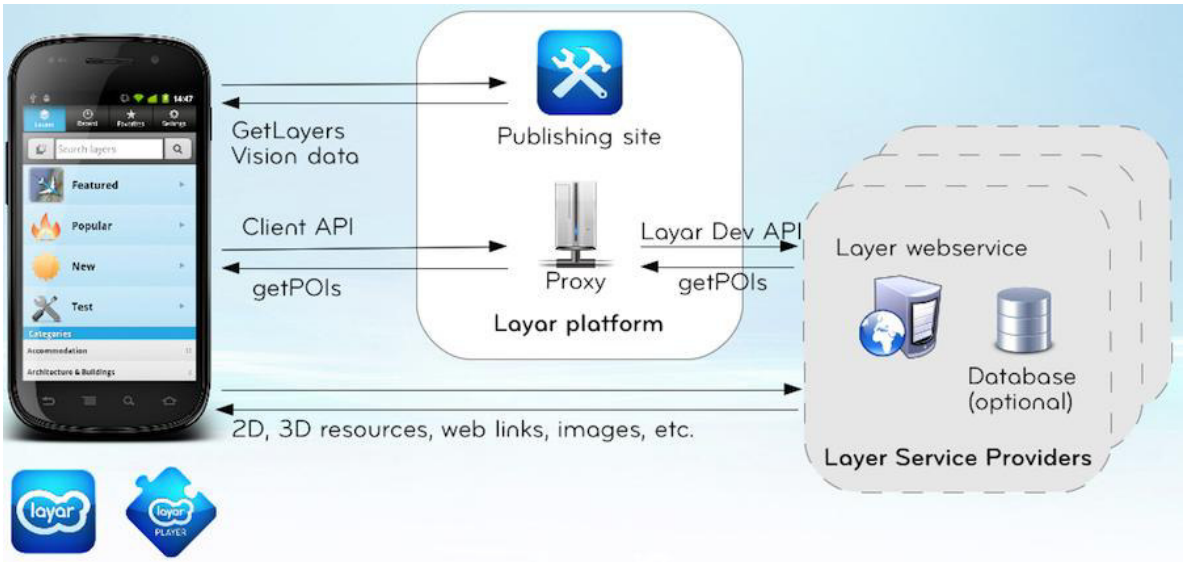


Figura 15 - Arquitectura de la plataforma Layar [45]

- **Navegador de realidad aumentada:** Es una aplicación móvil para sistemas operativos Android y iOS, que permite visualizar contenido de realidad aumentada, ya sea basada en la visión o en la geolocalización.

- **El servidor Layar:** Es un servidor que provee interfaces de comunicación para el navegador, el sitio web de publicación de contenidos y para la capa de proveedores de servicio.

- **Sitio web de publicación:** Es una plataforma web que permite la administración de todo el ciclo de vida de una aplicación Layar. Ofrece funcionalidades para la creación, edición, prueba, publicación, mantenimiento y terminación de una capa Layar.

- **Capa de proveedores de servicio:** Esta capa está orientada a la coordinación entre una aplicación Layar y desarrollos de terceros. Dicha comunicación se hace a través de servicios web.

- **Capa de recursos:** Esta capa está íntimamente relacionada con la capa de proveedores de servicios. Sin embargo, esta capa la función de esta capa es de proveer el contenido a ser visualizado en el navegador de realidad aumentada.

Sobre las APIs que maneja la plataforma tenemos:

- **API cliente de Layar:** Gestiona la comunicación entre el servidor Layar y la aplicación Layar. Esta API no está abierta al público.

- **API de desarrolladores:** Es la interfaz de comunicación entre el servidor Layar y la capa de proveedores de servicio. Abierta a los desarrolladores. Permite crear, publicar una aplicación Layar y registrarla en los proveedores de servicio. Actualmente esta API está basada en la arquitectura REST.

3.1.4 Wikitude

Wikitude fue la primera empresa en sacar al mercado una plataforma para el desarrollo de aplicaciones de realidad aumentada. Es la empresa que lleva el liderazgo en el esfuerzo de estandarización de esta tecnología junto con la *Open Geospatial Consortium (OGC)* desarrollando el lenguaje ARML 2.0.

Al igual que Layar, Wikitude tiene una plataforma y un SDK para la creación de aplicaciones de realidad aumentada basadas en la geolocalización y en el reconocimiento. Actualmente brinda soporte para Android, iOS y *Smartglasses*.

Wikitude se ha caracterizado por la incorporación de múltiples tecnologías para crear aplicaciones en su plataforma, así, incluye la posibilidad de utilizar APIs nativas y en JavaScript para los sistemas que soporta. También incluye una serie de plugins que permiten utilizar su SDK con otras tecnologías como Xamarin y Unity. Su arquitectura puede verse en la Figura 16 [16]

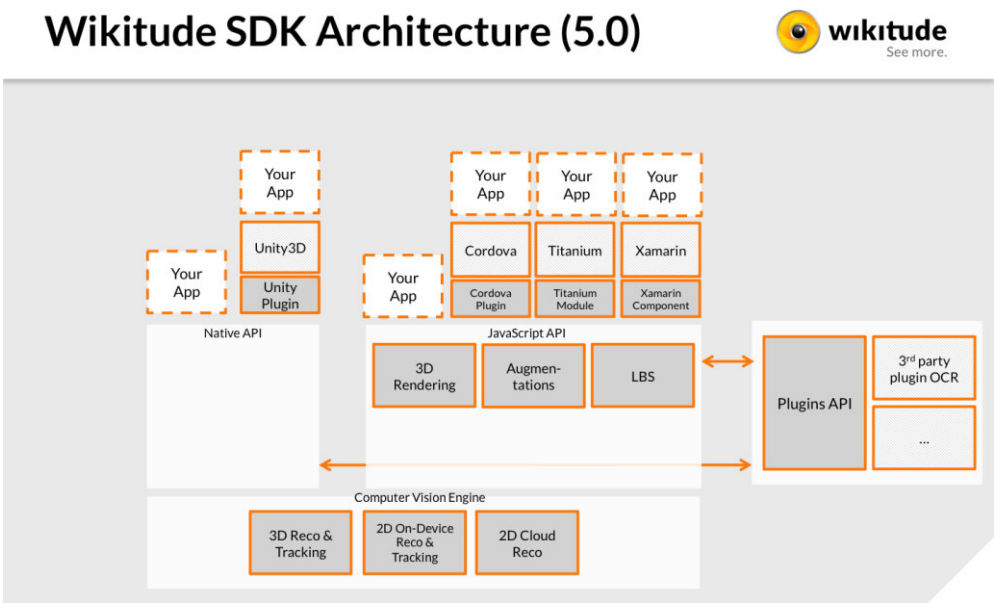


Figura 16- Arquitectura de Wikitude [16]

Para el caso de aplicaciones de realidad aumentada basadas en la visión Wikitude provee un servicio Cloud que mediante servicios REST permite a las aplicaciones basadas en su SDK reconocer las imágenes involucradas en dicha aplicación [46].

3.1.5 Vuforia

Vuforia es una plataforma que permite la creación de aplicaciones de realidad aumentada mediante el reconocimiento de imágenes, objetos, texto y marcadores. En la actualidad brinda soporte para los sistemas operativos Android y iOS, adicionalmente tiene soporte para Unity [47].

La plataforma tiene tres componentes básicos:

- **Motor Vuforia:** Es una biblioteca para los sistemas operativos Android y iOS que permite la creación de aplicaciones de realidad aumentada y permite administrar el reconocimiento de imágenes, la utilización de la cámara y la visualización de los objetos aumentados. Vuforia brinda un SDK para acceder a los servicios de esta biblioteca [47]. La arquitectura general puede verse en la Figura 17.

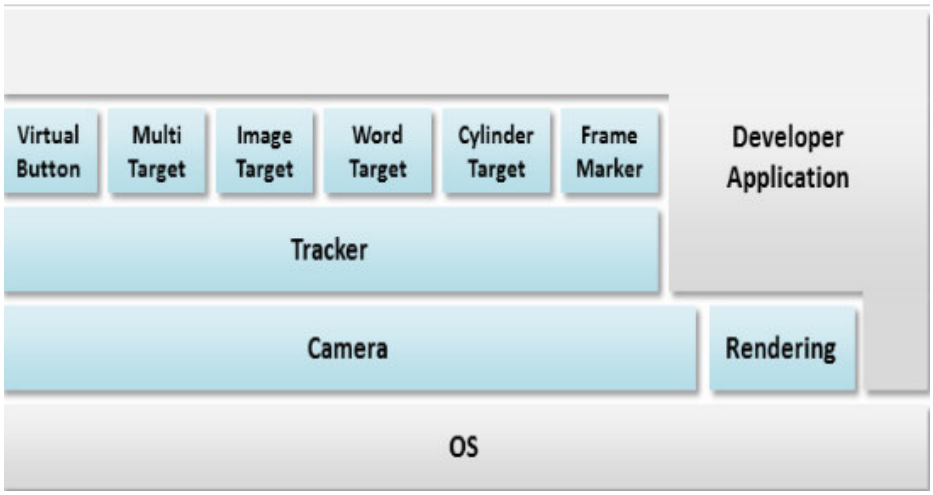


Figura 17 - Arquitectura del framework Vuforia [48]

- **Herramientas:** Son aplicaciones que permiten crear bases de datos de las imágenes a ser reconocidas y administrar las licencias para el uso del SDK [47].

- **Servicios de reconocimiento cloud:** Que permite administrar grandes bases de datos de imágenes mediantes el uso de servicios web [47].

3.1.6 Comparación de los navegadores y sistemas de realidad aumentada

La comparación de los navegadores y sistemas de realidad aumentada, se realizará tomando en cuenta cinco parámetros diferentes:

- **SPRT - Soporte:** Se realizará una comparación en dos ámbitos de soporte. La comparación se realizó tomando como base lo estudiado en [49]:

    - **TPAR - Tipo de realidad aumentada que permite:** Se consideran los siguientes tipos:

        - **VS**: Para aplicaciones basadas en la visión
        - **GL**: Para aplicaciones basadas en la localización.

    - **TPEL - Tipos de elementos aumentados soportados:** Se consideran los siguientes tipos:

        - **IMG:** Imágenes en diferentes formatos (JPG, PNG, GIF)
        - **TXT:** Texto simple
        - **HTM:** Páginas creadas en HTML.
        - **3DE:** Elementos 3D

- **SCNS - Descripción de escenas:** Se realizará una comparación de acuerdo al lenguaje utilizado para describir las escenas de realidad aumentada:

    - **LN:** Indica si el navegador utiliza algún lenguaje para describir las escenas de realidad aumentada.
    - **ST:** Indica si el lenguaje utilizado es estándar o no.

- **ARQU - Arquitectura:** Se realizará una comparación de acuerdo a la arquitectura sobre la cual se basa el navegador o plataforma. Las comparaciones se tomaron de lo estudiado en [31].

    - **GW :** Arquitectura Gateway

- **PF:** Arquitectura Plataforma
- **WB:** Arquitectura Web
- **ST:** Arquitectura Standalone

- **INOP - Interoperabilidad:** Se evalúa si el navegador realidad aumentada permite la interoperabilidad con otros sistemas semejantes. La comparación se realizó tomando como base lo estudiado en [49].

  Se definen los siguientes valores:

  - **VS:** Interoperabilidad en las aplicaciones basadas en la visión.
  - **GL:** Interoperabilidad en las aplicaciones basadas en la geolocalización.

La Tabla 5 muestra el resultado de la comparación de los diferentes navegadores y plataformas de realidad aumentada, en base a los parámetros señalados.

AR	SPRT						SCNS		ARQU				INOP	
	TPAR		TPEL											
	VS	GL	IMG	TXT	HTM	3DE	LN	ST	GW	PF	WB	ST	VS	GL
MARW	X	X	X		X				X			X		
ARGON	X	X	X		X		X					X		
Layar	X	X	X		X	X	X*	X*		X		X		X*
Wikitude	X	X	X		X	X	X*	X*		X		X		X*
Vuforia	X	X	X			X				X		X		

Tabla 5 - Comparación de los navegadores y sistemas de realidad aumentada

(*) Se realizó una prueba de concepto usando el estándar ARML2.0 para hacer interoperables sus servicios de realidad aumentada basados en la geolocalización. Sin embargo no se tiene información si esta prueba se encuentra actualmente en producción [23].

3.2 Bibliotecas para el seguimiento de marcadores de realidad aumentada

En este subcapítulo se describen algunas bibliotecas existentes para la creación de aplicaciones de realidad aumentada basadas en la visión.

3.2.1 ARToolKit

ARToolKit es una biblioteca de código abierto que permite la creación de aplicaciones de realidad aumentada en dispositivos móviles como Android, iOS y Windows Phone, y aplicaciones de escritorio como Windows, Linux y OS X. Además posee un *plugin* que permite su integración con el motor de videojuegos Unity. Actualmente se encuentra en la versión 5.3.1 [50].

Inicialmente presentado en 1999 [13], ARToolKit ha evolucionado en los últimos años como una de las bibliotecas más importantes para la creación de aplicaciones de realidad aumentada. En el 2001 fue liberado como un proyecto de código abierto por la empresa ARToolworks y a terceras partes crearon versiones para otras plataformas como Flash (FLARToolKit), Processing (NyARToolKit) y JavaScript (JSARToolKit). En el 2015, ARToolworks fue comprado por DAQRI que libera el SDK de ARToolKit bajo la licencia LGPL [51].

La versión para Android del framework incluye una librería en C++ llamada ARWrapper y una librería en Java llamada ARBaseLib que permite la creación de aplicaciones de realidad aumentada basadas en la visión. ARBaseLib se comunica con ARWrapper mediante JNI, ver Figura 18, gestiona la creación y manejo de la cámara para dispositivos Android y crea una superficie de OpenGL ES en sobre la cual se pueden renderizar gráficos, tal y como se ve en la Figura 19 [52].

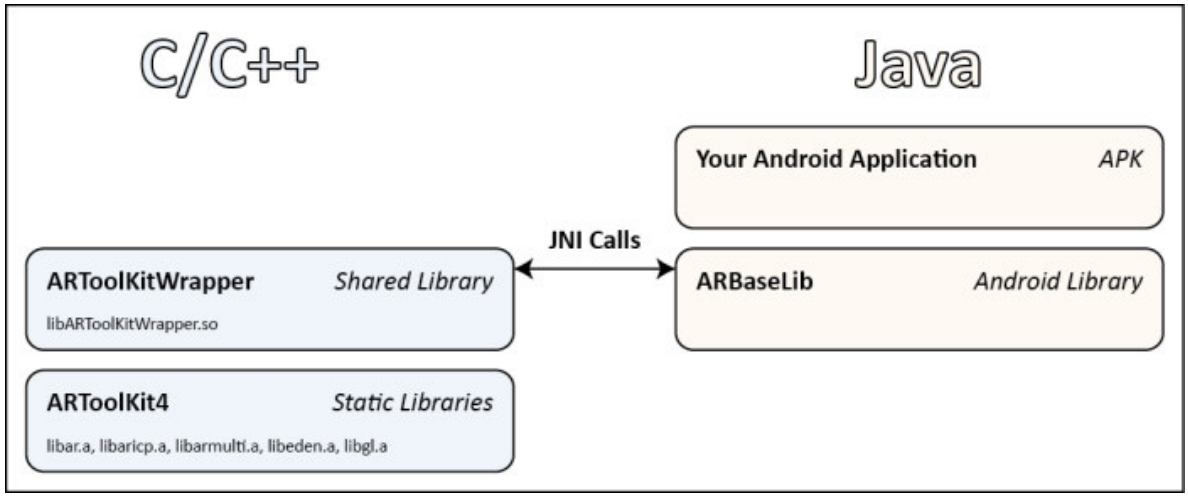


Figura 18 - Comunicación entre ARBaseLib y ARWrapper [52]

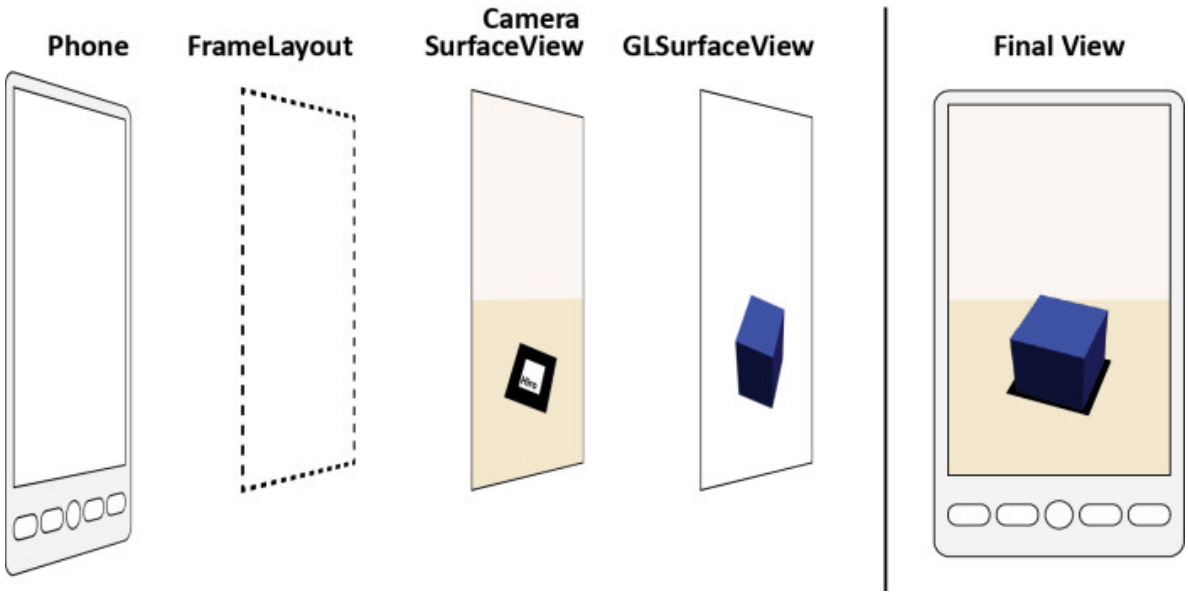


Figura 19 - Estructura en capas de una aplicación con ARBaseLib [52]

3.2.2 Alvar

Alvar es una biblioteca que permite la creación de aplicaciones de realidad aumentada y realidad virtual. Fue creada por el VTT Technical Research Centre of Finland. La última versión liberada fue la 2.0 en Mayo del 2012 [14].

Esta biblioteca permite la creación de aplicaciones para Windows y Linux y tiene dependencia con la biblioteca de visión computacional OpenCV. Sin embargo, es independiente de cualquier librería gráfica para el renderizado de objetos aumentados [14].

Actualmente permite la creación de aplicaciones de realidad aumentada basadas en la visión utilizando marcadores e imágenes (*markerless augmented reality*). Posee herramientas que ayudan a calibrar la cámara, esconder los marcadores de la vista, entre otros [14].

3.2.3 BazAr

BazAr es una biblioteca de visión computacional creada por el Laboratorio de Visión Computacional del Instituto Politécnico Federal de Lausanne.

Permite ubicar puntos de detección en una imagen y hacer la comparación de dichos puntos con los de otra (Figura 20), de manera tal de que se pueda identificar si dichas imágenes son las mismas, esta tecnología permite realizar aplicaciones de realidad aumentada basadas en la visión sin el uso de marcadores.

Actualmente está disponible para sistemas operativos de escritorio Windows y Linux. Tiene dependencia con OpenCV [53].

Figura 20- Detección de puntos en dos imágenes con BazAr [53]

3.2.4 DroidAr

Es un proyecto de código abierto que permite la creación de aplicaciones de realidad aumentada basadas en la geolocalización y aplicaciones basadas en la visión usando marcadores. Actualmente se encuentra en la versión 2 y se encuentra licenciado bajo GNU GPL v3[54].

Esta biblioteca tiene dependencia con la biblioteca de visión computacional OpenCV y utiliza el estándar OpenGL para mostrar los objetos aumentados.

Sin embargo el proyecto no muestra actividad desde el 2013, y no se tiene claro si su desarrollo continúa [55].

3.2.5 Comparación de bibliotecas de realidad aumentada

Se tomarán en cuenta los siguiente parámetros como puntos de comparación:

- **OSYSAR - Sistemas operativos disponibles:** Indica los sistemas operativos para los cuales se puede desarrollar. La comparación se realizó tomando como base lo estudiado en [49].

  Se consideran los siguientes:
  - **L: Linux**
  - **W: Windows**
  - **X: OS X**
  - **A: Android**
  - **I: iOS**

- **MARBAR - Soporte para realidad aumentada basada en marcadores:** Indica si se permite la creación de aplicaciones de realidad aumentada basada en marcadores. La comparación se realizó tomando como base lo estudiado en [49].

- **MLESSAR - Soporte para realidad aumentada con marcadores naturales:** Indica si se permite la creación de aplicaciones de realidad aumentada basada en marcadores naturales (markerless augmented reality). La comparación se realizó tomando como base lo estudiado en [49].

- **GEOBAR - Soporte para realidad aumentada basada en la geolocalización:** Indica si el framework permite realizar aplicaciones de realidad aumentada

basadas en la geolocalización. La comparación se realizó tomando como base lo estudiado en [49].

- **OPENAR - Libertad de modificación:** Indica si la biblioteca es de código abierto y permite realizar modificaciones a su código fuente. La comparación se realizó tomando como base lo estudiado en [49].

- **SOPOAR - Soporte :** Indica el tipo de soporte dado por la organización creadora de la biblioteca. La comparación se realizó tomando como base lo estudiado en [49].
  Se definen los siguientes tipos:

  - **F - Full:** Indica amplio soporte para desarrolladores, mediante listas de correo, foros activos, tutoriales claros, conferencias, FAQ.

  - **M - Moderado:** Indica un soporte moderado para desarrolladores, puede contener uno o más de los elementos indicados en la categoría Full.

  - **L - Limitado:** Soporte limitado, sólo se indica un medio de contacto. Ej: correo electrónico.

  - **D - Desconocido:** No se conoce el soporte brindado.

La Tabla 6 presenta el resultado de la comparación realizada tomando en cuenta los parámetros descritos. El símbolo 'x' denota la presencia del indicador.

Biblioteca	OSYSAR					MARBAR	MLESSAR	GEOBAR	OPENAR	SOPOAR
	L	W	X	A	I					
ARToolKit	x	X	x	x	x	x	x		x	M
Alvar	x	X		x	x	x	x		x*	L
BazAr	x	X					x		x	D

DroidAr				x		x		X	x	L

Tabla 6- Comparación de bibliotecas de realidad aumentada

(*) Alvar es un proyecto de código abierto únicamente en su versión para los sistemas operativos Linux y Windows.

De la comparación anterior, se visualiza entonces que Artoolkit, Alvar y DroidAr son las tres bibliotecas que cumplen con los requisitos mínimos necesarios para desarrollar el presente trabajo, dichos requisitos son: soporte para Android, soporte de realidad aumentada basada en marcadores, libertad de modificación, y documentación de soporte para su uso. A continuación se profundiza el estudio para estas bibliotecas:

- **MARGEN - Generación de marcadores:** Este parámetro indica qué tipo de herramientas brinda la biblioteca para la generación de nuevos marcadores de realidad aumentada personalizados. La comparación se realizó tomando como base lo estudiado en [49].

  Se definen los siguientes tipos:

  - **ON:** Herramientas Online
  - **OFF:** Herramientas Offline. Ej: Scripts

- **MARTYP - Tipos de marcadores soportados:** Este parámetro indica cuáles son los marcadores soportados por la biblioteca. La comparación se realizó tomando como base lo estudiado en [49].

  Definimos dos tipos:

  - **S - Marcadores Únicos:** Indica si la biblioteca es capaz de reconocer sólo un marcador por cada frame de cámara.

- **M - Marcadores Múltiples:** Indica si la biblioteca es capaz de reconocer más de un marcador por cada frame de cámara.

- **DEPREC - Dependencia con otras bibliotecas:** Indica si la biblioteca tiene dependencia con otras bibliotecas externas para realizar el reconocimiento de marcadores.

- **RENDER - Tecnología de objetos aumentados:** Indica la tecnología que utiliza la biblioteca para poder mostrar objetos aumentados.

La Tabla 7 muestra el resultado de la comparación realizada.

BIBLIOTECA	MARGEN		MARTYP		DEPREC	RENDER
	ON	OFF	S	M		
ARToolKit	x	x	x	x	Independiente	Independiente
Alvar		x	x	x	OpenCV	OpenGL OpenSceneGraph
DroidAr		x	x		OpenCV	OpenGL

Tabla 7 - Comparación de bibliotecas para realidad aumentada basada en la visión

CAPÍTULO IV
NAVEGADOR DE REALIDAD AUMENTADA

En este capítulo se documenta el desarrollo del navegador de realidad aumentada. Se comienza realizando una introducción a la metodología utilizada y su aplicación para el presente trabajo. A continuación, se documenta el proceso de desarrollo seguido y se realiza una descripción de los componentes involucrados, también se muestran los resultados de las pruebas de validación del estándar y finalmente, se presenta una descripción de tipo funcional del navegador desarrollado.

4.1 Metodología

El desarrollo del presente trabajo se realizó aplicando los principios del framework Scrum. A continuación se detallan los principios de esta metodología, los roles, los artefactos y los eventos/ceremonias que se llevan a cabo.

4.1.1 Scrum

Scrum es un *framework* que permite a los equipos solucionar problemas complejos y producir productos de alto valor [56]. Es una metodología ágil, iterativa e incremental, donde "los proyectos son desarrollados mediantes iteraciones denominadas *sprints*. Cada *sprint* constituye un conjunto de procesos de bajo nivel del ciclo de vida del software desde la fase de planificación hasta la fase de pruebas."[57]

Scrum no es un proceso estandarizado, donde se siguen pasos previamente definidos, es más bien un "marco de trabajo para organizar y administrar trabajo"[58] que utiliza principios, valores y prácticas que una organización puede tomar y adaptar a sus necesidades.

Esta metodología está basada en tres pilares que se definen a continuación:

- **Transparencia:** El proceso es visible a todos los involucrados. Todos los participantes siguen los estándares definidos.
- **Inspección:** Se revisan constantemente los artefactos y el progreso de acuerdo al objetivo definido para el Sprint.

- **Adaptación:** Si se detectan desviaciones en los procesos más allá de los límites permitidos, se realizan los ajustes necesarios en dichos procesos.

Scrum es un marco de trabajo enfocado a las personas, que promueve valores tales como honestidad, franqueza, coraje, respeto, atención, confianza, empoderamiento y colaboración. Además, se caracteriza por definir prácticas que engloban un conjunto de roles, actividades y artefactos [58]. Cada uno de estos elementos se describe a continuación.

ROLES

"Los equipos basados en Scrum están constituidos por tres roles diferentes: Product Owner, Scrum Master y el equipo de desarrollo. Aunque pueden existir otros roles en un proyecto, el framework sólo requiere estos tres" [58]. A continuación se describen cada uno de los roles.

- **Product Owner: "**Es el responsable de maximizar el valor del producto y del trabajo realizado por el equipo de desarrollo" [59]. Entre sus responsabilidades se encuentran:
    - Expresar claramente los ítems del Product Backlog
    - Organizar los elementos del Product Backlog de acuerdo a los objetivos
    - Optimizar el valor del trabajo realizado por el equipo de desarrollo
    - Asegurar que el Product Backlog es visible, claro y transparente para todos.
    - Asegurar que el equipo de desarrollo entiende los ítems del Product Backlog.

- **Scrum Master:** Es líder del equipo de desarrollo y se asegura que todos los participantes entiendan y acojan las prácticas, principios y valores de Scrum. Es un facilitador y toma acciones para resolver problemas que puedan interferir con el desarrollo. También ayuda y da soporte al equipo en el proceso de administración de cambios al adoptar esta metodología [58].

- **Equipo de Desarrollo:** El equipo de desarrollo es aquel grupo de personas que se encargan de llevar el trabajo a cabo durante cada Sprint. Algunas de las características que definen al equipo son [59]:
    - Se auto - organizan
    - Cumplen múltiples roles
    - No existen sub-equipos
    - Sus miembros tienen diversas especializaciones

ACTIVIDADES

Scrum define un conjunto de actividades cuyo objetivo es tratar de minimizar y regularizar la necesidad de reuniones no definidas en el framework. Cada una de estas actividades ofrece además la oportunidad de inspeccionar y adaptar algún proceso. Además de ser una ventana donde se promueve la transparencia crítica. A continuación se describen las actividades:

- **Sprint**: Es un periodo de tiempo cuya duración es de un mes o menos, en el cual se desarrolla un producto terminado, que es potencialmente utilizable por el usuario. Se genera un incremento en el valor del producto.
- **Sprint Planning:** Se define el trabajo a realizar durante el Sprint.
- **Daily Scrum:** Es una reunión de 15 minutos que el equipo de desarrollo realiza de manera diaria, y donde se responden las siguientes preguntas:
    - ¿Qué se realizó el día de ayer para alcanzar el objetivo del Sprint?
    - ¿Qué se realizará el día de hoy?
    - ¿Qué impedimentos tengo que pueden evitar alcanzar el objetivo?
- **Sprint Review:** Es una reunión que se lleva a cabo al finalizar el Sprint para validar el incremento y la adaptación al Product Backlog y el trabajo realizado durante el Sprint.

  Como resultado de esta reunión se define un probable Product Backlog para el siguiente Sprint.

- **Sprint Retrospective:** Es una reunión donde el equipo se reúne, se evalúa y crea un plan para mejorar en el siguiente Sprint.

ARTEFACTOS

Los artefactos definidos en Scrum representan una ventana a la transparencia y permiten al equipo evaluarse y adaptarse [59]. Se definen los siguientes:

- **Product Backlog:** Es una lista ordenada de todo lo que debe contener el producto a desarrollar. Es el lugar donde se documentan los cambios al proyecto. Es dinámico y se va actualizando a medida que se va teniendo mejor conocimiento de los requerimientos [59].

- **Sprint Backlog:** Es un conjunto de elementos del Product Backlog que serán trabajados durante el Sprint. El Sprint Backlog es dinámico, en la medida de que si es requerido, se añade trabajo para alcanzar el objetivo definido en el Sprint y a medida que se va completando el trabajo se va actualizando la estimación del trabajo restante [59].

4.2 Desarrollo del navegador de realidad aumentada

En este subcapítulo se documenta el desarrollo del navegador de realidad aumentada mediante la metodología Scrum. Se comenzará describiendo una etapa previa de planificación y se continuará con el desarrollo del trabajo realizado en cada uno de los *sprints*.

A continuación se presentará la documentación del desarrollo, detallando los componentes que se incluyen en el navegador de realidad aumentada y describiendo el proceso que se utiliza para el reconocimiento y presentación de una escena de realidad aumentada. Finalmente se presenta el resultado de las pruebas de la validación del estándar.

4.2.1 Planificación

En esta etapa de planificación se han definido algunas características básicas del desarrollo del navegador de realidad aumentada. Esta etapa comienza seleccionando las herramientas que se utilizarán para el desarrollo, tales como la biblioteca de realidad aumentada y la definición de la arquitectura de alto nivel del navegador.

Posteriormente, se define el alcance del estándar ARML2.0 y se seleccionan las pruebas de validación del estándar en base al alcance definido.

Finalmente, se crean las historias de usuario y el Product Backlog para el desarrollo. La Tabla 8 muestra un resumen de estas actividades. Posteriormente se explican cada una de ellas en detalle.

Planificación	
Actividad	**Sub actividad**
Selección de herramientas	Biblioteca de seguimiento de marcadores
	Arquitectura de aplicación
Definición del alcance	Elementos de ARML 2.0
	Pruebas de validación ARML 2.0
Creación del Product Backlog	

Tabla 8.- Actividades de planificación para el desarrollo

4.2.1.1 Selección de las Herramientas

Selección de la biblioteca para el seguimiento de marcadores

La selección de una biblioteca para el seguimiento y reconocimiento de marcadores es de vital importancia en una aplicación de realidad aumentada. Para el presente trabajo, se han considerado tres aspectos importantes para la elección de la herramienta:

- **Soporte de marcadores:** Capacidad para utilizar marcadores no naturales de realidad aumentada, y realizar el registro de más de un marcador para el reconocimiento.

- **Soporte para el sistema operativo Android:** Tomando en cuenta si la biblioteca puede usar en el sistema operativo Android.

- **Posibilidad de modificación:** Donde se evalúa si la biblioteca permite la modificación de sus interfaces y métodos de visualización de los objetos aumentados de acuerdo a las necesidades del proyecto.

Tomando en cuenta estos tres criterios, y en base a la comparaciones realizadas en la Tabla 6 y la Tabla 7 de la sección 3.2.5 se define que la biblioteca a utilizar será ARToolKit.

Selección de la arquitectura de realidad aumentada

Los sistemas descritos en la sección 2 fueron comparados de manera arquitectónica usando el modelo de referencia de T. Reicher y la clasificación basada en dicho modelo por B. Butchart (secciones 2.1.5 y 2.1.6) . Estos sistemas están basados en diferentes tipos de arquitecturas de realidad aumentada, algunos de ellos, tales como Wikitude, Layar y Junaio, constituyen una plataforma que implementan dos modelos arquitectónicos.

Sin embargo, una característica básica de todos los navegadores y sistemas estudiados es que implementan el modelo arquitectónico *Standalone*. Como se estudió anteriormente, este modelo agrupa todos los subsistemas de realidad aumentada definidos en el modelo de referencia dentro de una aplicación cliente. Dicho cliente puede conectarse a la web para recibir actualizaciones sobre el contenido.

Tomando estos antecedentes como referencia, se define el modelo arquitectónico *Standalone* como modelo base para el desarrollo del navegador de realidad aumentada del presente trabajo.

4.2.1.2 Definición del alcance

Selección de los elementos del estándar ARML 2.0

Como se ha estudiado anteriormente, no todos los elementos del lenguaje ARML2.0 están diseñados para describir escenas para aplicaciones basadas en la visión. Es por ello, que para el presente trabajo se han seleccionado sólo los elementos que permiten describir una escena de este tipo. La Tabla 9 muestra los elementos y atributos seleccionados.

Elemento	Atributo	Descripción del atributo
ARElement	Id	Identificador del elemento
Anchor	enabled	Permite describir si un Anchor está habilitado o no, afectando el resultado final.
ARAnchor	assets	Conjunto de elementos aumentados relacionados a un Anchor.
Tracker	uri	Contiene información en formato binario del marcador de realidad aumentada.
TrackableConfig	tracker	Identificador que relaciona a un *Trackable* con un *Tracker*
	Src	Enlace al marcador de realidad aumentada. Permite la interoperabilidad entre navegadores
	Order	Define el orden en el cual las configuraciones *TrackableConfig* se tomarán en cuenta
Trackable	config	Configuración del elemento Trackable.
VisualAsset	enabled	Permite describir si el elemento aumentado está habilitado o no.
	zOrder	Orden de presentación de los elementos aumentados.
	Orientation	Rotación aplicada a los elementos aumentados.
Label	href	Enlace a una URL de una página descrita en HTML.
	src	Código HTML en línea.
	viewportWidth	Tamaño del contenido de una página HTML.
Text	src	Texto que se aumentará

	style	Estilo aplicado al texto, en términos de color de fuente y color de fondo
Image	href	Enlace que apunta a la imagen.

Tabla 9.- Elementos de ARML2.0 dentro del alcance de la implementación

Selección de las pruebas de la implementación descriptiva

Se evaluará la implementación del navegador de realidad aumentada basado en el estándar ARML 2.0 usando las pruebas de conformidad de la implementación descriptiva que este lenguaje define de acuerdo a los elementos seleccionados dentro del alcance de la implementación.

La Tabla 10 se presenta las pruebas que se utilizarán:

Pruebas de conformidad sobre la implementación descriptiva		
Código	**Propósito**	**Prueba**
2.1	Validar que la implementación reconoce archivos ARML2.0.	Verificar que la implementación reconoce archivos ARML2.0 correctos.
2.3	Validar que la implementación no permite ARElements con un id user.	Verificar que la implementación ignora ARElements con id = user
2.5	Validar que la implementación ignora un Anchor y sus objetos asociados cuando este es deshabilitado	Verificar que la implementación ignora cualquier Anchor y sus VissualAssets asociados cuando su propiedad enable tiene valor false.
2.6	Validar que la implementación reconoce Anchors que no tienen un Feature asociado	Verificar que la implementación añade Anchors a la escena.
2.18	Validar que la implementación no falla en caso el Tracker es desconocido.	Verificar que la implementación ignora cualquier Tracker los Trackables asociados desconocidos.

2.24	Validar que la implementación asigna correctamente el valor de orden por defecto de un TrackableConfig, en caso este no se encuentre asignado.	Verificar que la implementación asigna un valor por defecto máximo para el atributo order en caso este no haya sido definido.
2.30	Validar que la implementación ignora un VisualAsset que está deshabilitado.	Verificar que la implementación ignora un VisualAsset cuando la propiedad enable se encuentra en false.
2.31	Validar que la implementación oculta objetos correctamente.	Verificar que la implementación oculta objetos de acuerdo a la distancia indicada en el valor Zorder
2.35	Validar que la implementación cumple las reglas de precedencia en un Label.	Verificar que la implementación brinda mayor precedencia al atributo src sobre el atributo href, en caso ambos tengan valor.
2.36	Validar que la implementación no falle en caso un Label sea inválido.	Verificar que la implementación ignora Labels que tienen los parámetros src y href (ambos) no definidos.
2.39	Validar que la implementación coloca correctamente el valor por defecto para el atributo viewportWidth de un Label	Verificar que la implementación coloca el valor de 256 cuando la propiedad viewportwidth no tiene valor o tiene un valor negativo.
2.40	Validar que la implementación reemplaza correctamente los metadatos de nombre y descripción	Verificar que la implementación reemplace los metadatos $[name] y $[description] con un cadena vacía, en caso el Label no esté relacionado a un Feature
2.41	Validar que la implementación reemplaza correctamente los metadatos.	Verificar que la implementación reemplace los metadatos con formato: $[(XPath-Expression)] con un cadena vacía, en caso el Label no esté relacionado a un Feature

2.43	Validar que la implementación reemplaza correctamente los metadatos de nombre y descripción.	Verificar que la implementación reemplaza cualquier ocurrencia de $[name] y $[description] con cadena vacía en caso el elemento Text no esté relacionado a ningún label.
2.44	Validar que la implementación reemplaza correctamente los metadatos colocados en el texto.	Verificar que la implementación reemplaza cualquier ocurrencia de $[(XPath-Expression)] con cadena vacía en caso el elemento Text no esté relacionado a ningún label.
2.45	Validar que la implementación coloca el valor por defecto correcto para un elemento Text.	Verificar que la implementación coloca el valor #000000 (negro) para el parámetro color del elemento Text.
2.46	Validar que la implementación coloca el valor por defecto correcto para el color de fondo un elemento Text.	Verificar que la implementación coloca el valor #000000 (transparente) para el parámetro background-color del elemento Text.
2.47	Validar que la implementación no falle en caso un elemento Image sea inválido.	Validar que la implementación ignora un elemento Image cuando no soporta el formato de la imagen.
2.56	Validar que la ejecución de la orientación manual de VisualAsset.	Verificar que la implementación ejecuta las rotaciones en el orden roll - tilt - heading.

Tabla 10.- Pruebas de conformidad sobre la implementación descriptiva

Fuente: Elaboración propia con información tomada de [32]

4.2.1.3 Product Backlog

Durante el periodo de planificación inicial, se identificaron 17 historias de usuario que definían las funcionalidades del navegador de realidad aumentada. El detalle de cada una de las historias de usuario, puede encontrarse en el ANEXO III, un resumen de las mismas puede encontrarse en la Tabla 11. Es importante destacar que se clasificaron las diferentes historias de usuario en base a la importancia dentro del alcance del presente trabajo de investigación.

PRODUCT BACKLOG			
Nro. de H.U.	**Título H.U.**	**Prioridad**	**Épica**
1	Reconocer escenas basadas en ARML2.0	ALTA	x
2	Obtener un archivo ARML2.0 desde una URL	ALTA	
3	Aumentar objetos tipo Texto	ALTA	
4	Aumentar objetos tipo Imagen	ALTA	
5	Registro de marcadores de realidad aumentada	ALTA	
6	Deshabilitar marcadores	MEDIA	
7	Habilitar marcadores	MEDIA	
8	Deshabilitar elementos aumentados	MEDIA	
9	Habilitar elementos aumentados	MEDIA	
10	Reconocer el color de texto de un elemento tipo Texto	MEDIA	
11	Reconocer el color de fondo de un elemento tipo Texto	MEDIA	
12	Uso de cámara posterior	ALTA	
13	Aumentar objetos tipo HTML	ALTA	
14	Tamaño de objetos HTML aumentados	MEDIA	
15	Orientación de objetos aumentados	MEDIA	
16	Orden de despliegue de objetos aumentados	MEDIA	
17	Presentación de objetos aumentados de acuerdo al marcador	ALTA	x

Tabla 11.- Product Backlog

4.2.2 Sprints

Para el desarrollo del presente proyecto se realizaron 5 iteraciones o *sprints*, cada uno con 3 semanas de duración. La Tabla 12 presenta un resumen de las iteraciones realizadas y los objetivos que se plantearon para cada una de ellas.

Nor. Sprint	OBJETIVO DEL SPRINT
1	Navegador capaz de utilizar la cámara posterior y obtener un archivo ARML2.0
2	Navegador capaz de realizar el registro de marcadores definidos en un archivo ARML2.0
3	Navegador capaz de realizar la el reconocimiento y la presentación de elementos tipo Texto definidos en un archivo ARML2.0
4	Navegador capaz de realizar la presentación de elementos tipo Imagen definidos en un archivo ARML2.0 y capaz de reconocer marcadores y objetos aumentados habilitados y deshabilitados.
5	Navegador capaz de realizar la presentación elementos tipo HTML definidos en un archivo ARML2.0

Tabla 12.- Descripción de las iteraciones en el proceso de desarrollo

4.2.3 Primer *Sprint*

En este *Sprint* se desarrolló la funcionalidad para la utilización de cámara y obtención de un archivo ARML2.0. La Tabla 13 muestra el sprint backlog preparado para esta iteración.

SPRINT BACKLOG			
Historia de Usuario	**Tarea**	**Estimación Inicial (en días)**	**Duración Real (en días)**
12	Despliegue de cámara	0.5	0.5
	Creación de eventos de cámara para inicio del reconocimiento	0.5	0.5
	Configuración automática de la cámara de acuerdo a la orientación del dispositivo móvil	0.5	0.5
	Interfaz de usuario previo al despliegue de cámara	0.5	0.5
2	Creación del administrador de descargas	2	2
	Creación de clases para la administración de tareas en un hilo secundario	3	3
	Administración de mensajes	3	5

	de mensajes entre el hilo secundario y el hilo principal		
	Control de errores producto de la descarga	2	2
	Interfaz de usuario del módulo de descargas	1	1
TOTAL		13	15

Tabla 13.- Sprint Backlog de la iteración 01

RESULTADO DEL SPRINT

Como producto del trabajo desarrollado se logró obtener el despliegue de la cámara y la descarga del archivo ARML2.0 a partir de la indicación del usuario. Este proceso, se puede ver en la Figura 21.

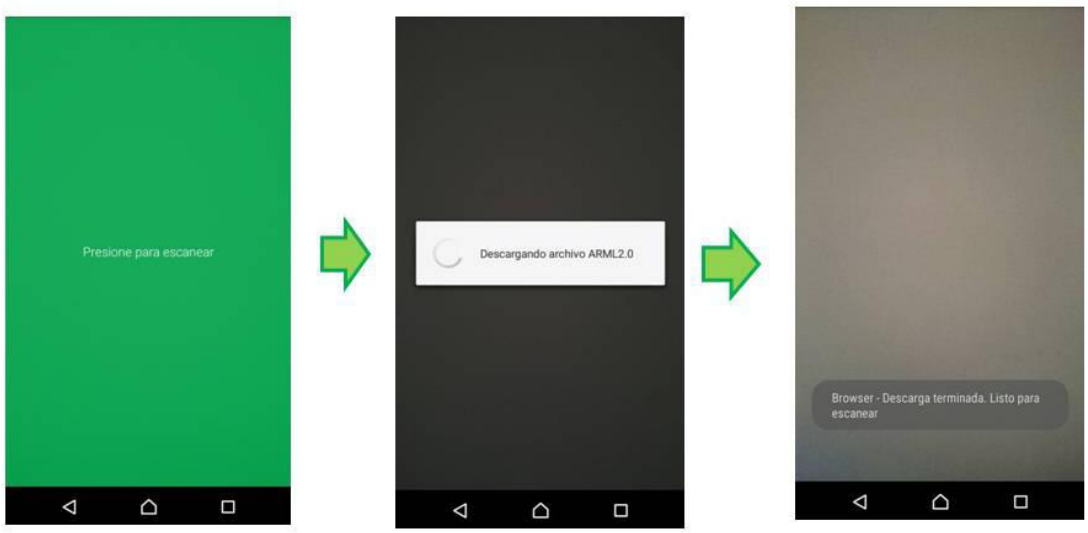


Figura 21.- Resultado del primer sprint

RETROSPECTIVA DEL SPRINT

En esta iteración, se observó una desviación de dos días en la ejecución del trabajo planificado. Esto se debió principalmente a que se tuvieron problemas con la forma de procesamiento de tareas de larga duración y la administración de hilos en el sistema operativo Android. Sin embargo, se lograron cumplir los objetivos planificados para el *Sprint*.

4.2.4 Segundo *Sprint*

Para el desarrollo de esta iteración, se planificó desarrollar la identificación y reconocimiento de marcadores de realidad aumentada; esta funcionalidad corresponde a las historias de usuario 1 y 5. Cabe mencionar que la historia de usuario 1, es una historia de usuario épica, cuya funcionalidad estará completa al final del proyecto.

Para la validación de esta iteración, se decidió colocar en pantalla una imagen predefinida, que se mostraba cuando el marcador era reconocido. Es importante resaltar que esta imagen no estaba especificada en el archivo ARML2.0. Adicionalmente se realizó la validación mediante el log de Android.

En la Tabla 14 se presenta el *sprint backlog* de esta iteración.

SPRINT BACKLOG			
Historia de Usuario	**Tarea**	**Estimación Inicial (en días)**	**Duración Real (en días)**
1 - 5	Integración de la biblioteca de realidad aumentada ARToolKit.	2	1
	Procesamiento del archivo ARML2.0	3	4
	Creación del árbol inicial de objetos ARML2.0	5	4
	Identificación de marcadores en el archivo ARML2.0.	3	4
	Descarga automática del archivo binario para reconocimiento de marcadores.	1	1
	Obtención del evento que reconoce un marcador de realidad aumentada.	2	1
TOTAL		15	15

Tabla 14.- Sprint *backlog* de la iteración 02

RESULTADO DEL SPRINT

La Figura 22 muestra el resultado obtenido en la segunda iteración. Se logró reconocer los marcadores de realidad aumentada definidos en un archivo ARML2.0.

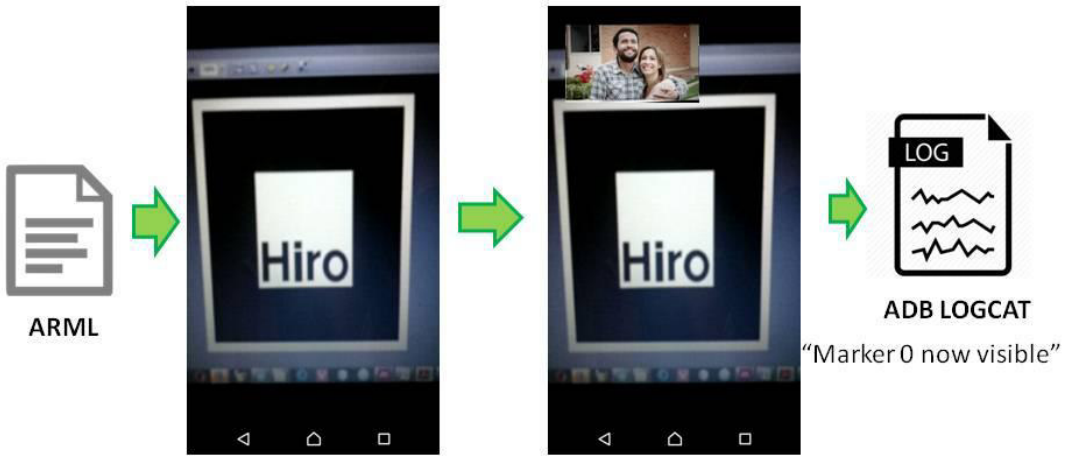


Figura 22.- Resultado del segundo sprint

Fuente: Elaboración propia

RETROSPECTIVA DEL SPRINT

Esta iteración logró cumplir satisfactoriamente los objetivos planteados. No hubo una desviación general en los días asignados al s*print*, pero si hubo diferencias en las días planificados para ciertas tareas.

4.2.5 Tercer *Sprint*

El objetivo de este *sprint* era desarrollar la funcionalidad que permitía aumentar objetos de tipo Texto e implementar las opciones para dar estilo a dicho elemento. Estas funcionalidades abarcaron el desarrollo de las historias de usuario 2 (historia épica), 3, 10 y 11.

El *sprint backlog* del esta iteración, puede verse en la Tabla 15.

SPRINT BACKLOG			
Historia de Usuario	**Tarea**	**Estimación Inicial (en días)**	**Duración Real (en días)**
17	Reconocimiento del elemento Text en un archivo ARML 2.0	1	1

1 - 3	Inclusión del elemento Text en el árbol de objetos arml	4	6
	Inclusión del elemento Text en el árbol de presentación de objetos aumentados	4	5
	Presentación de elementos tipo Text	2	2
	Asociación del elemento Text con un marcador de realidad aumentada	3	4
10	Reconocimiento de la propiedad *background-color*	0.5	0.25
11	Reconocimiento de la propiedad *font-color*	0.5	0,25
TOTAL		15	18

Tabla 15.- Sprint backlog de la iteración 03

RESULTADO DEL SPRINT

La Figura 23 muestra el resultado obtenido en el tercer *sprint*. Se logró reconocer los elementos tipo *Text* asociados a un marcador de realidad aumentada. También se implementaron las propiedades de estilo *background*-color y *font-color* para dicho elemento.

RETROSPECTIVA DEL SPRINT

Durante el desarrollo de este s*print*, se presentaron problemas y retrasos en las tareas involucradas con el reconocimiento y la presentación del elemento *Text*. Esto se debió principalmente a errores presentes en la sincronización de algunas tareas que se ejecutaban en el hilo secundario de la aplicación. Dichos errores requirieron un esfuerzo de refactorización de algunas estructuras de datos y de funcionalidad relacionada a los marcadores.

A pesar de estos inconvenientes, se lograron terminar las funcionalidades para el *sprint*

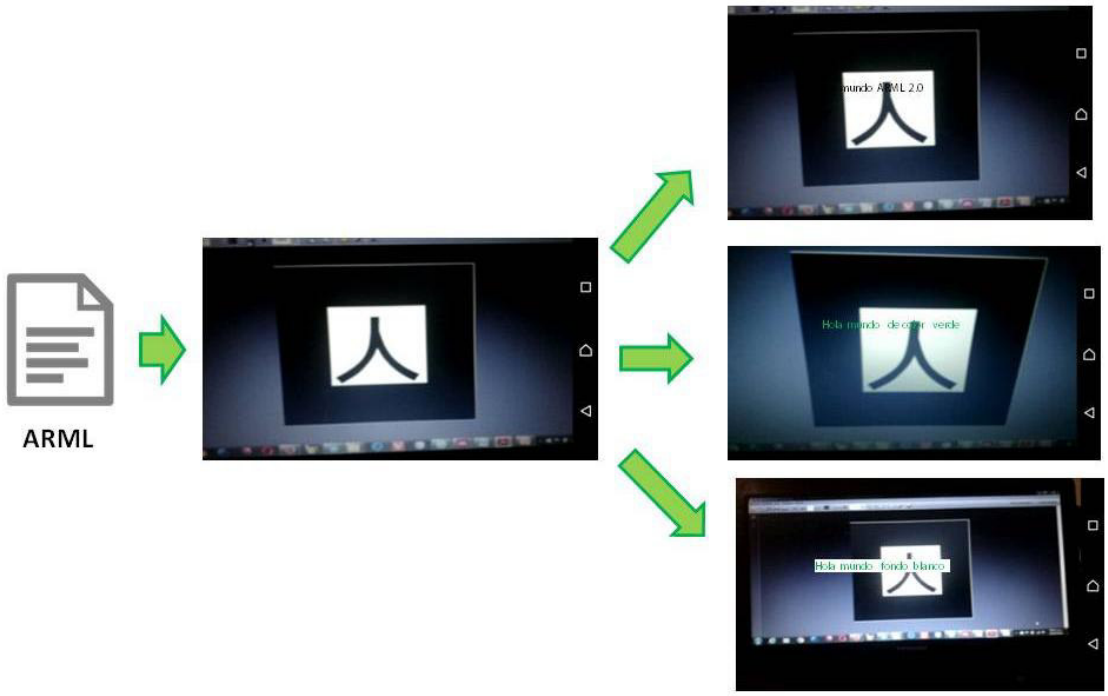


Figura 23.- Resultados del tercer sprint

4.2.6 Cuarto *Sprint*

Para esta iteración se planificó realizar la implementación de los objetos aumentados tipo imagen y realizar el reconocimiento de marcadores y objetos aumentados deshabilitados y habilitados en el archivo ARML2.0. Estas funcionalidades corresponden a las historias de usuario 1, 4, 6, 7,8 y 9.

La Tabla 16 muestra el *sprint backlog* de esta iteración.

SPRINT BACKLOG			
Historia de Usuario	**Tarea**	**Estimación Inicial (en días)**	**Duración Real (en días)**
17	Reconocimiento del elemento *Image* en un archivo ARML 2.0	1	1
1-4	Inclusión del elemento *Image* en el árbol de objetos arml	4	2

	Inclusión del elemento *Image* en el árbol de presentación de objetos aumentados	2	2
	Presentación de elementos tipo *Image*	2	2
	Asociación del elemento *Image* con un marcador de realidad aumentada	3	3
6	Funcionalidad para habilitar por defecto un marcador	0.5	0.5
	Habilitar un marcador mediante el atributo *enabled*	0.5	0.5
7	Deshabilitar un marcador mediante el atributo *enabled*	0.5	0.5
	Deshabilitar un todos los objetos aumentados relacionados al marcador	0.5	0.5
8	Deshabilitar un objeto aumentado mediante el atributo *enabled*	0.5	0.5
9	Habilitar un objeto aumentado mediante por defecto y mediante el atributo *enabled.*	0.5	0.5
TOTAL		15	13

Tabla 16.- Sprint backlog de la iteración 04

RESULTADO DEL SPRINT

La Figura 24 muestra el resultado obtenido en el cuarto *sprint*. Se logró reconocer los elementos tipo *Image* asociados a un marcador de realidad aumentada. También se implementó la funcionalidad que permite habilitar y deshabilitar marcadores y objetos aumentados.

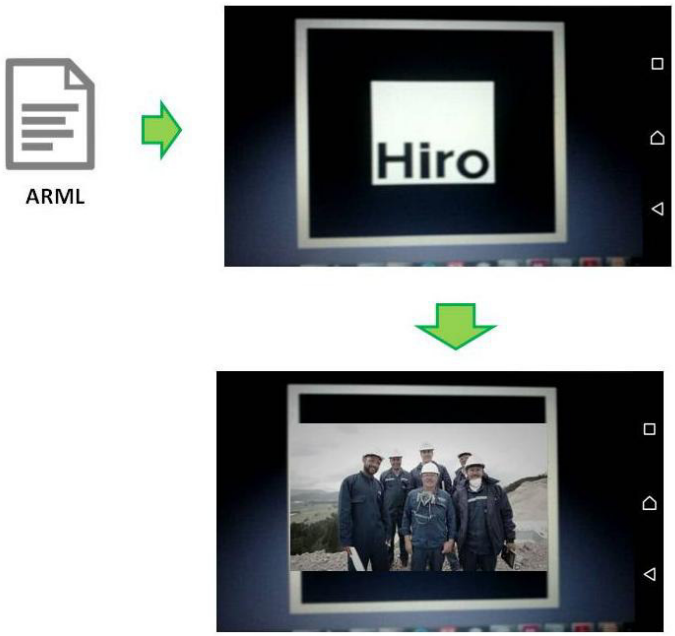


Figura 24.- Resultado del cuarto sprint

RETROSPECTIVA DEL SPRINT

El desarrollo de este s*print*, se realizó de manera más rápida que lo estimado debido a que la funcionalidad para aumentar imágenes es muy similar a la funcionalidad para aumentar texto que se desarrolló en la iteración anterior.

4.2.7 Quinto *Sprint*

En este *sprint* se implementó la funcionalidad correspondiente a los objetos aumentados de tipo *Label,* definida en la historia de usuario 13. Este desarrollo también involucra a la historia de usuario 1.

La Tabla 17 muestra el *sprint backlog* de esta iteración.

SPRINT BACKLOG			
Historia de Usuario	**Tarea**	**Estimación Inicial (en días)**	**Duración Real (en días)**
17	Reconocimiento del elemento *Label* en un archivo ARML 2.0	2	2
1 - 13	Inclusión del elemento *Label* en el árbol de objetos arml	4	3
	Inclusión del elemento *Label* en el árbol de presentación de objetos aumentados	3	3
	Presentación de elementos tipo *Label*	2	5
	Asociación del elemento *Label* con un marcador de realidad aumentada	2	3
14	Cambiar el tamaño del elemento HTML de acuerdo al viewport	1	1
15	Desplegar elementos aumentados de acuerdo a su zOrder	0.5	0.5
16	Posición de los elementos aumentados de acuerdo a su ángulo de rotación	0.5	0.5
TOTAL		15	18

Tabla 17.- Sprint backlog de la quinta iteración

RESULTADO DEL SPRINT

La Figura 25 muestra el resultado obtenido en el cuarto *sprint*. Se logró reconocer los elementos tipo *Image* asociados a un marcador de realidad aumentada. También se implementaron la funcionalidad de habilitación y des habilitación de marcadores y objetos aumentados correctamente.

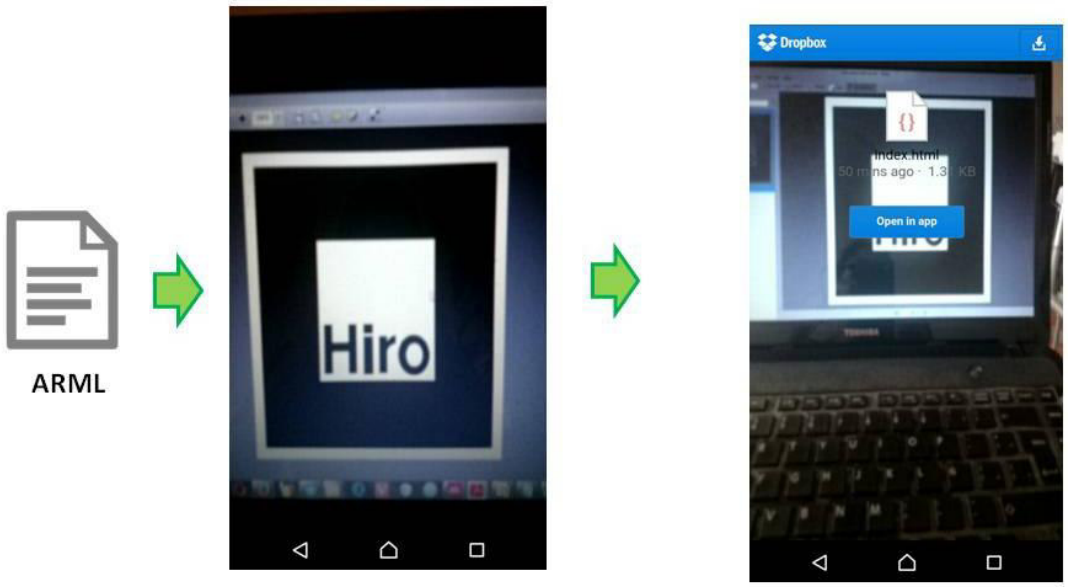


Figura 25.- Resultados del quinto sprint

RETROSPECTIVA DEL SPRINT

El desarrollo de este s*print* tuvo retrasos de acuerdo al tiempo estimado debido a que la funcionalidad para el módulo de presentación de los elementos tipo *Label* era nueva y difería bastante con respecto a la presentación de los objetos tipo *Text* e *Image.*

4.2.8 Documentación de la implementación

A continuación se presenta documentación básica que explica la implementación del navegador de realidad aumentada realizado.

4.2.8.1 Arquitectura

El navegador desarrollado está basado en la arquitectura *Standalone* de sistemas de realidad aumentada. Como se estudió en la sección 2.1.6, en este tipo arquitectónico todos los subsistemas del navegador se encuentran dentro de la misma aplicación.

La Figura 26 muestra la arquitectura de alto nivel del navegador desarrollado. Se distinguen tres módulos: Interfaz de usuario, Procesamiento y Aplicación cliente. A continuación se describen:

Módulo de Interfaz de usuario: Este módulo se encarga de gestionar los elementos necesarios para la presentación de escenas de realidad aumentada, y contiene el subsistema *Presentación* de la arquitectura estándar, lo que permite la representación de cada uno de los elementos en una vista del navegador según su tipo.

Contiene además, un administrador de interfaz de usuario que permite la comunicación con el subsistema de procesamiento mediante mensajes. Todas las actividades relacionadas con este módulo se ejecutan en el hilo principal de la aplicación Android.

Módulo de Procesamiento: Este módulo concentra todos los subsistemas que se utilizan para el procesamiento de escenas de realidad aumentada. Haciendo un paralelo con los subsistemas de la arquitectura estándar, contiene además los subsistemas de *Interacción*, *Tracking*, *Contexto* y *Mundo Modelo*.

Adicionalmente consta de un administrador general, que coordina las actividades de estos subsistemas con el módulo de Interfaz de usuario mediante mensajes. Todas las actividades relacionadas con este módulo se ejecutan en un hilo secundario.

Aplicación cliente: Este módulo permite la extensión de un navegador de realidad aumentada, añadiendo funcionalidad para algún propósito particular.

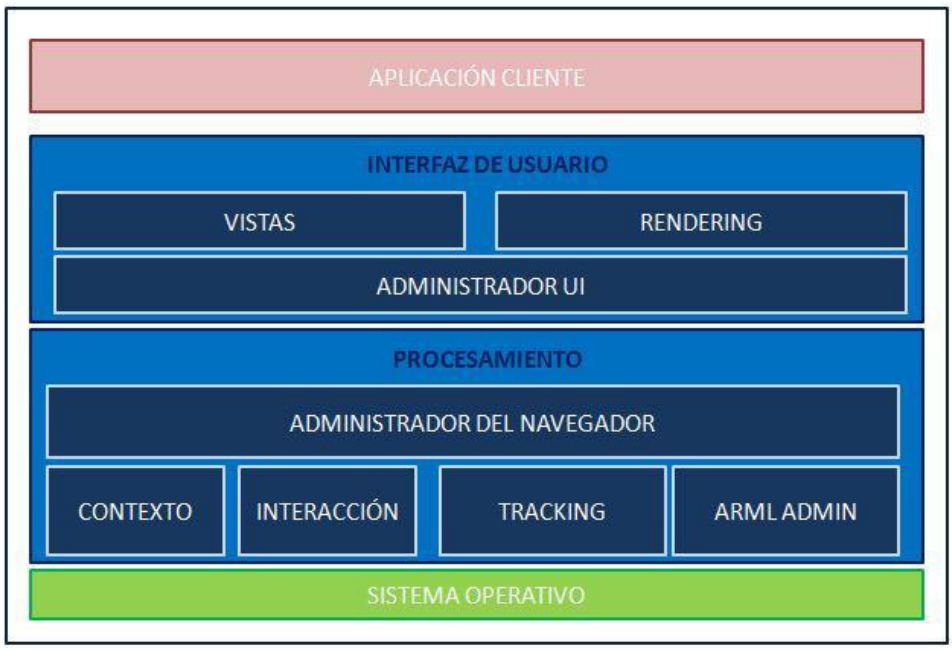


Figura 26.- Arquitectura del navegador de realidad aumentada

Fuente: Elaboración propia

La comunicación entre el módulo de interfaz de usuario y el de procesamiento, se realiza mediante el envío de mensajes por parte de cada uno de los administradores de los módulos. Para ello, se realizó una variación de la implementación del patrón *Active Object* [60] con las herramientas que provee el sistema operativo Android por medio de *Handlers* y *HandlerThreads*.

De esta manera, dentro del hilo principal de la aplicación del módulo de Interfaz de usuario, se creó la clase *UIHandler* que recibe todos los mensajes que se muestran en la interfaz así como los elementos de una escena de realidad aumentada. Estos mensajes se apilan en una estructura de datos provista por el sistema operativo.

El módulo de procesamiento ejecuta todas las tareas de reconocimiento y procesamiento de las escenas de realidad aumentada de manera secuencial en un hilo secundario, este hilo es creado mediante la clase *FlowController*, esta clase hace uso de una pila interna donde se alojarán todos los mensajes recibidos. Al mismo tiempo, hace uso de la clase *FCHandler*, que se encarga de administrar todas los mensajes recibidos en la pila, ordenar su ejecución y enviar mensajes al *UIHandler*. El comportamiento descrito se puede visualizar con el diagrama de secuencia de la Figura 27.

Cabe mencionar, que este mecanismo de comunicación permite dos tipos de tareas:

Tareas de ejecución: Este tipo de tareas se ejecutan cuando el sistema operativo les asigna tiempo de ejecución. Implementan la interfaz *Runnable*.

Mensajes Simples: Son mensajes que son utilizados para transmitir únicamente una acción al controlador. No transmiten datos.

Dentro del desarrollo este tipo de mensajes se envían por ejemplo, cuando ocurre un error interno dentro de una tarea en ejecución, donde no es posible para la tarea corregir el estado y continuar con el flujo.

Por ejemplo, si dentro de una tarea de descarga de archivos, se pierde la conexión a internet el controlador del módulo de procesamiento, enviará un mensaje al controlador de la UI indicando que apile el mensaje con código *INTERNET_CONNECTION_PROBLEMS.* De manera tal, que cuando el procesador asigne tiempo de ejecución al hilo de la UI y llegue el momento de ejecutar dicho mensaje, el controlador *UIHandler* realizará una acción de acuerdo al mensaje recibido (mostrar un mensaje de error en pantalla).

Es importante destacar que el modelo elegido brinda las siguiente ventajas [61]:

- Creación de tareas independientes que pueden ser reusadas
- Aseguramiento de ejecución secuencial de las tareas
- Permite tener puntos de decisión entre cada tarea, lo que permite cambiar el flujo en base a los resultados.
- Define un mecanismo para pasar datos entre tareas, evitando el acoplamiento entre clases.

Finalmente, la Tabla 18, la Tabla 19 y la Tabla 20 presentan una descripción general de las clases descritas anteriormente con los atributos y métodos más importantes.

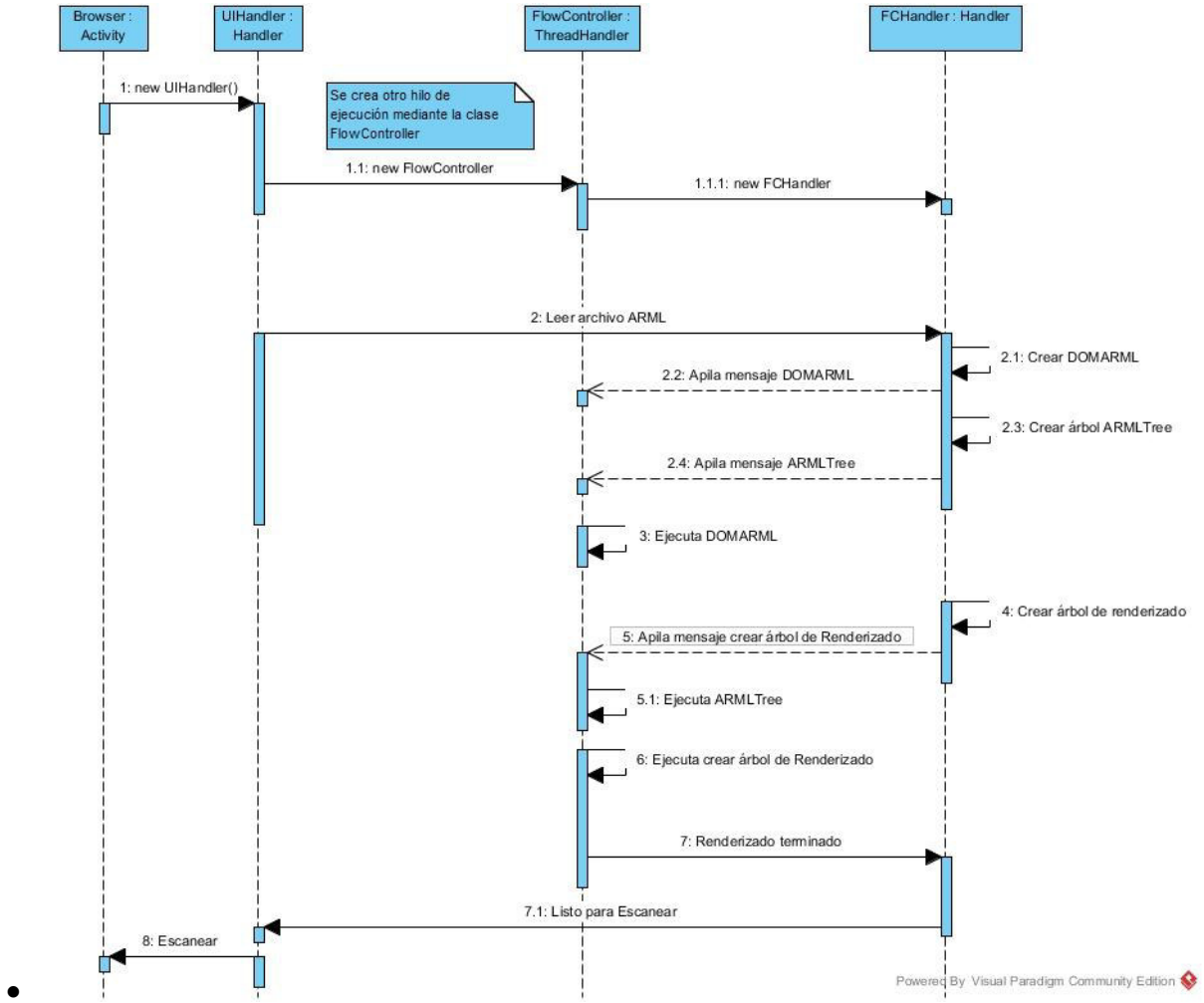


Figura 27.- Esquema de comunicación del navegador de realidad aumentada

Clase	**UIHandler**
Super clase	Handler
Descripción	Responsable de ejecución de las tareas en la interfaz de usuario.
ATRIBUTOS	
Contexto	El contexto de la aplicación. Necesario para la creación de vistas.
MÉTODOS	
handleMessage	Toma un mensaje de la pila de ejecución y define el comportamiento a realizar de acuerdo al tipo.

Tabla 18.- Descripción de la clase UIHandler

Clase	FCHandler
Superclase	Handler
Descripción	Responsable de ejecución de las tareas del módulo de procesamiento.
ATRIBUTOS	
Contexto	El contexto de la aplicación. Necesario para la creación de vistas.
MÉTODOS	
handleMessage	Toma un mensaje de la pila de ejecución y define el comportamiento a realizar de acuerdo al tipo.

Tabla 19.- Descripción de la clase FCHandler

Clase	FlowController
Superclase	HandlerThread
Descripción	Crear un hilo de ejecución con una pila para mensajes para el módulo de procesamiento
ATRIBUTOS	
uiHandler	Referencia al UIHandler
backgroundHandler	Referencia al FCHandler
MÉTODOS	
OnLooperPrepared	Crea la instancia del FCHandler

Tabla 20.- Descripción de la clase FlowController

4.2.8.2 Subsistema Mundo Modelo

Tal como se estudió anteriormente, el subsistema Mundo Modelo provee acceso a la representación del mundo real y el mundo virtual, así como a las relaciones existentes entre estos dos mundos. Para el caso del presente trabajo, estas relaciones se encuentran en los archivos que describen las escenas de realidad aumentada basados en ARML2.0.

El modelado a nivel de objetos de software de una escena de realidad aumentada, se realiza en dos pasos:

1. Lectura de la estructura del documento ARML
2. Creación de un árbol basado en el modelo de objetos ARML

En el primer paso, el archivo ARML es leído y se crea su estructura DOM [62], esto nos brinda una estructura de datos inicial para procesarlo. A continuación, se realiza un lectura de cada nodo, utilizando una estrategia de búsqueda en profundidad.

A medida que se va leyendo la información de los nodos, se crea un árbol de objetos ARML, donde el nodo raíz está representado por objetos de tipo *ARMLElement* y los nodos hijos son de dos tipos: *Tracker* y *Trackable*. Los nodos de tipo *VisualAsset* y *Anchor*, son asignados al elemento *Trackable* correspondiente, según las relaciones expuestas en el archivo; de no encontrarse alguna relación para estos elementos, estos son ignorados de acuerdo a lo indicado por el estándar.

A continuación se presenta un ejemplo del mecanismo descrito anteriormente. La Figura 28 muestra un escena de realidad aumentada basada en ARML2.0, en ella se describe un elemento *Trackable* que tiene un *VisualAsset* de tipo *Text* . Dicho *Trackable* se encuentra relacionado a un marcador con identificador *#hiromarker*. La descripción de dicho marcador se encuentra en elemento *Tracker*.

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Text>
          <src>HOLA MUNDO ARML 2.0</src>
          <style>
              font-color:#000000;
              background-color: #00000000;
          </style>
        </Text>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="http://www.anvicordova.com/unmsm/android/artoolkit/single" />
      <!--Proprietary information about the tracker -->
      <src xlink:href="http://www.anvicordova.com/unmsm/android/artoolkit/patt.hiro" />
    </Tracker>
  </ARElements>
</arml>
```

Figura 28.- Ejemplo de archivo ARML2.0

La lectura e interpretación de este archivo comienza cuando el controlador del módulo de procesamiento (*FCHandler)*, recibe el mensaje *BUILD_ARML_TREE*, en este instante, se crea una instancia de la clase *XMLReader* que utiliza librerías estándar de java para crear la estructura DOM del archivo. Para el caso de nuestro ejemplo, se crearía la estructura definida en la Figura 29 .

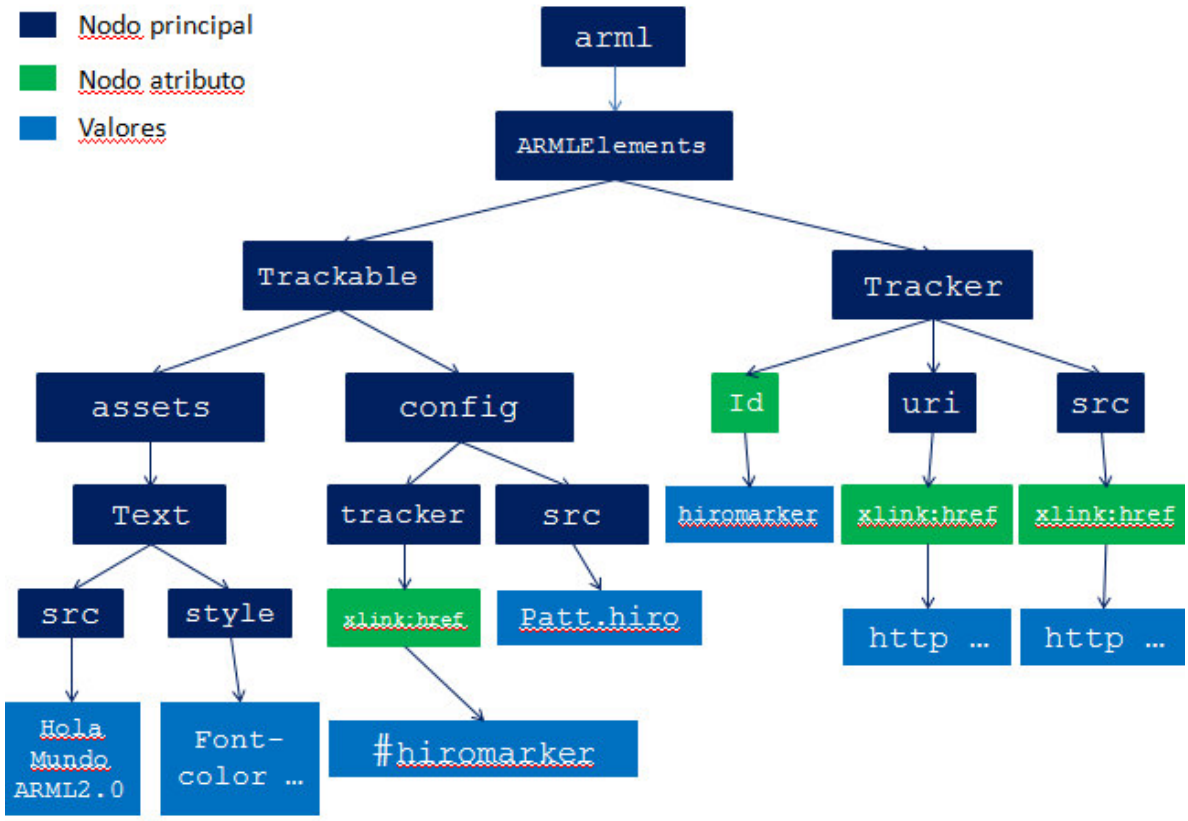


Figura 29.- Estructura DOM de un archivo ARML2.0

Una vez creada esta estructura, el controlador general del módulo de procesamiento *FCHandler*, apila una tarea de ejecución que creará un árbol de objetos ARML, basado en el Diagrama de Objetos del estándar (Figura 10).

Una vez que el procesador asigna tiempo de ejecución a la tarea, se realiza un recorrido en profundidad por la estructura DOM creando un árbol de objetos ARML (clase *ARMLTree)* mediante la clase *ARMLTreeGenerator*. Para el ejemplo en estudio, la estructura creada puede verse en la Figura 30 .

Visualizamos entonces que obtenemos una estructura mucho más simple, con un árbol cuya raíz es un elemento *ARMLElement* genérico y cuyos nodos hijos son objetos ARML2.0. Adicionalmente, esta nueva estructura conserva las relaciones especificadas en la escena, esto se evidencia en la relación de composición entre el elemento *Trackable* y el elemento *Text*.

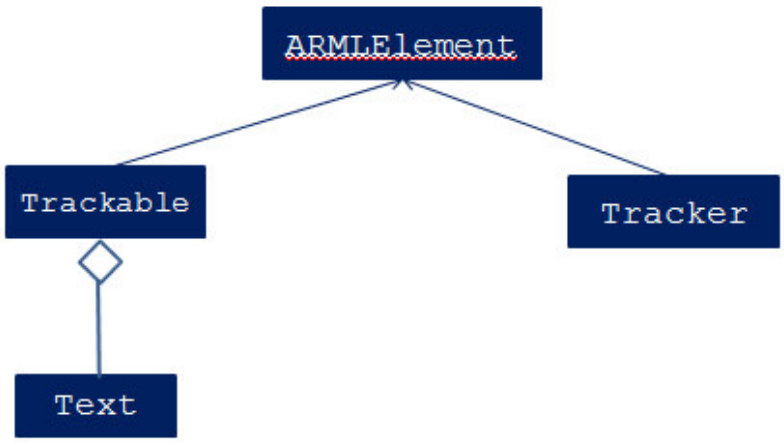


Figura 30.- Ejemplo de árbol de objetos ARML2.0

El proceso anteriormente descrito se encuentra plasmado en el diagrama de secuencia de la Figura 31; la Figura 32 muestra el diagrama de clases del subsistema Mundo Modelo.

Finalmente, la Tabla 21, Tabla 22 y Tabla 23 documentan las clases más importantes de este subsistema.

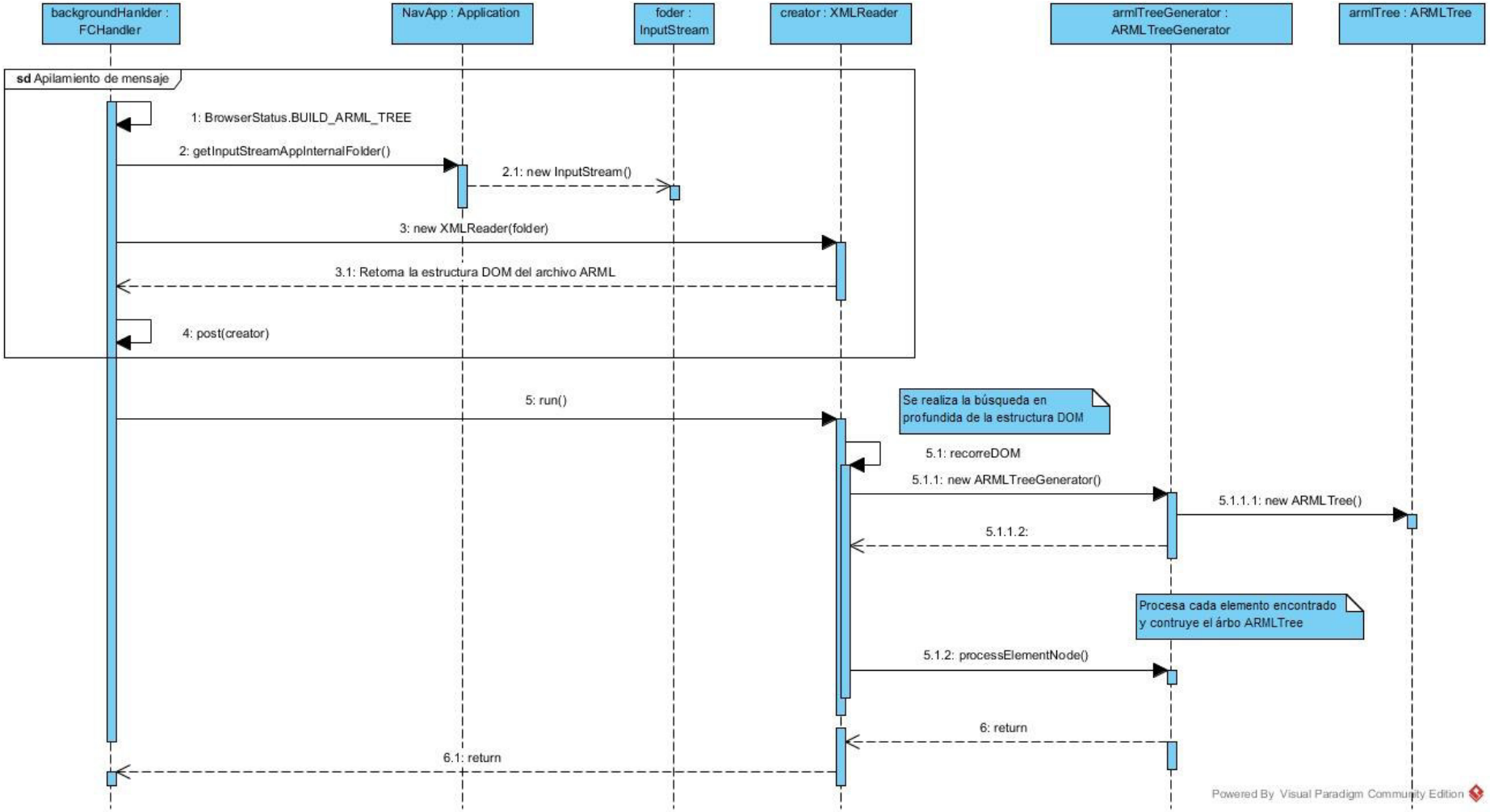


Figura 31.- Diagrama de secuencia Mundo Model

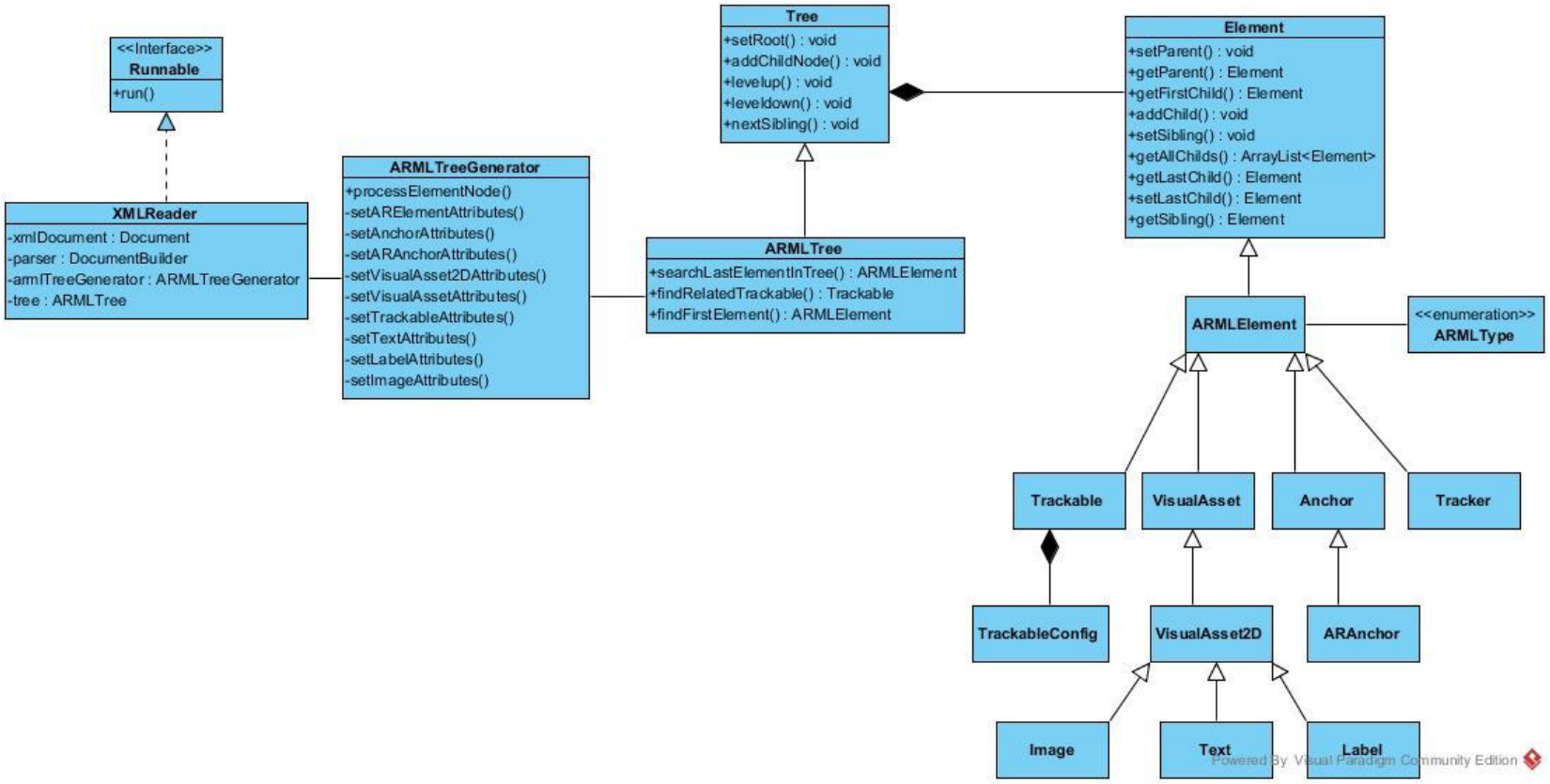


Figura 32.- Diagrama de clases del subsistema Mundo Modelo

Clase	XMLReader
Implementa	Runnable
Descripción	Representa una tarea de ejecución que permite el recorrido en profundidad de la estructura DOM del archivo ARML
ATRIBUTOS	
xmlDocument	Referencia al documento XML
Parser	Permite obtener la estructura DOM y recorrerla.
MÉTODOS	
Run	Ejecución de la tarea
RecorreDOM	Permite iniciar la búsqueda en profundidad

Tabla 21.- Descripción de la clase XMLReader

Clase	ARMLTreeGenerator
Descripción	Genera un árbol de objetos ARML
ATRIBUTOS	
Tree	Instancia a el árbol ARML que se quiere crear
MÉTODOS	
processElementNode	Procesa un Elemento de la estructura DOM
SetARElementAttributes	Asigna los atributos de un nodo tipo ARElement
setAnchorAttributes	Asigna los atributos de un nodo tipo Anchor
setARAnchorAttributes	Asigna los atributos de un nodo tipo ARAnchor
setVisualAsset2DAttributes	Asigna los atributos de un nodo tipo VisualAsset2D
setVisualAssetAttributes	Asigna los atributos de un nodo tipo VisualAsset
setTrackableAttributes	Asigna los atributos de un nodo tipo Trackable
setTextAttributes	Asigna los atributos de un nodo tipo Text
setLabelAttributes	Asigna los atributos de un nodo tipo Label
setImageAttributes	Asigna los atributos de un nodo tipo Image

Tabla 22.- Descripción de la clase ARMLTreeGenerator

Clase	ARMLTree
Superclase	Tree
Descripción	Representación de un árbol ARML2.0
MÉTODOS	
SearchLastElementInTree	Busca el último elemento añadido al árbol de un tipo determinado
findRelatedTrackable	Dado un ID busca los elementos Trackables asociados
findFirstElement	Busca el primer elemento de un tipo dentro del árbol

Tabla 23.- Descripción de la clase ARMLTree

4.2.8.3 Subsistema de Seguimiento

Este subsistema es el encargado de realizar el registro, reconocimiento y seguimiento del marcador de realidad aumentada. Estas dos actividades se realizan en dos momentos diferentes con ayuda de la biblioteca ARToolKit. A continuación se presenta el mecanismo implementado.

REGISTRO DEL MARCADOR DE REALIDAD AUMENTADA

El estándar ARML2.0 define el elemento *Tracker* como el elemento donde se describe la información sobre los marcadores que manifiestan la conexión entre el mundo real y el mundo virtual. Este elemento distingue dos atributos: el atributo *src* que permite colocar información propietaria del marcador en formato binario, y el atributo *uri* que permite colocar información general que describe el marcador para que otras implementaciones del estándar puedan interpretar la escena descrita.

Para el caso de ARToolKit, el marcador se representa como un archivo binario, que es generado en un proceso anterior al reconocimiento mediante herramientas del framework. Por lo cual dentro de la escena de realidad aumentada se utilizará el atributo *src* para colocar una URL que permita la descarga de dicha información.

Una vez se creado el árbol de objetos ARML descrito en la sección 4.2.8.2, el navegador se prepara para interpretar la escena. Para ello, el controlador del módulo de procesamiento *FCHandler* apila un mensaje denominado *CONFIGURE_SCENE*, dicho mensaje hace uso de la clase *SceneConfigurator* que realiza una consulta al árbol de objetos ARML y obtiene una lista con todos los elementos de tipo *Tracker* registrados en este; a continuación, se hace un recorrido de esta lista y se registra cada uno de los

marcadores encontrados (con el atributo *src*). El framework retornará el ID de cada marcador correctamente registrado.

Para el ejemplo en estudio propuesto en la Figura 28, el elemento a registrar sería el que se encuentra en la dirección http://www.anvicordova.com/unmsm/android/artoolkit/patt.hiro.

El proceso descrito anteriormente, puede verse en el diagrama de secuencia de la Figura 33.

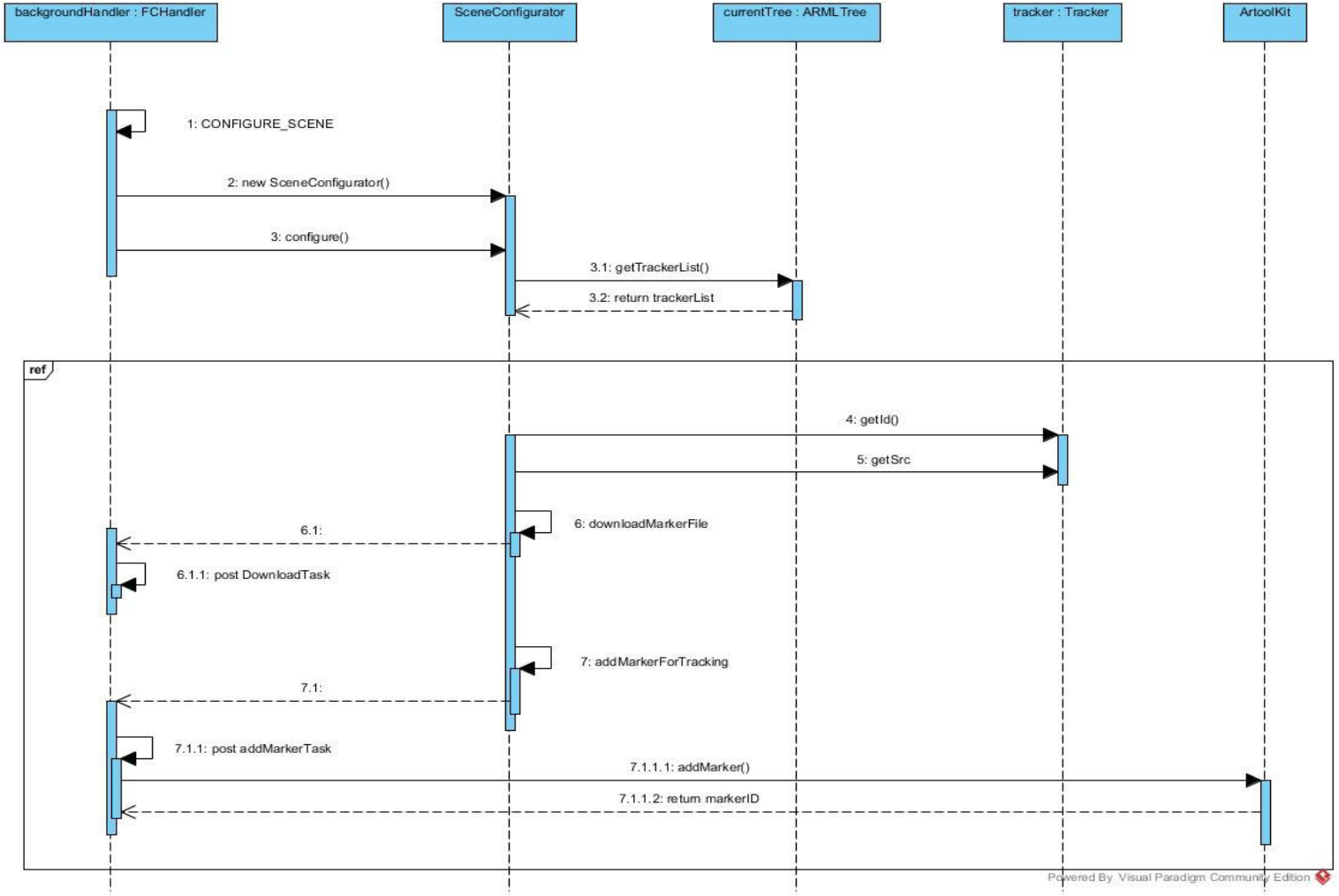


Figura 33.- Diagrama de secuencia para registro de marcadores de realidad aumentada

RECONOCIMIENTO Y SEGUIMIENTO DE MARCADORES DE REALIDAD AUMENTADA

El reconocimiento y seguimiento de los marcadores es realizado por el *framework* ARToolKit. Esta biblioteca nos brinda una interfaz Java, que nos permite verificar por cada frame de cámara, si un marcador previamente registrado se encuentra o no presente. Esto es realizado mediante la sentencia:

```
ARToolKit.getInstance().queryMarkerVisible(key)
```

Esta sentencia toma como parámetro un valor entero que corresponde al identificador del marcador. Este valor es generado automáticamente por el *framework* al momento del registro como hemos visto en la sección anterior, y retorna un valor booleano que nos permite saber si el marcador con ID *key* se encuentra visible .

4.2.8.4 Subsistema de Interacción

Según lo estudiado en la sección 2.1.5, este subsistema se encarga de recoger la información sobre eventos que indique el usuario, tales voz o gestos. Sin embargo, el estándar actualmente no define mecanismos para este tipo de interacciones.

4.2.8.5 Subsistema de Presentación

Este subsistema permite la presentación de los elementos del mundo virtual descritos en una escena de ARML2.0 en pantalla. Este proceso tiene 3 fases:

1. Definición de las vistas de presentación de objetos virtuales.
2. Preparación del árbol de objetos virtuales por cada marcador registrado.
3. Presentación en pantalla de los objetos virtuales cuando un marcador es reconocido.

A continuación se explican cada una de estas fases:

VISTAS PARA PRESENTACIÓN DE OBJETOS VIRTUALES

La presentación de objetos virtuales en pantalla, se realiza mediante el apilamiento de vistas, utilizando las herramientas que provee el sistema operativo Android, para ello se hace uso de los elementos *FrameLayout* y *RelativeLayout.*

La Figura 34 muestra la disposición anteriormente definida. La vista *FrameLayout* actúa como una especie de contenedor donde se ubican las siguientes vistas:

- *CameraView*: Que se encarga del manejo de la cámara del dispositivo móvil.
- *RelativeLayout:* Donde se despliegan los elementos tipo *Image, Text y Label* de ARML2.0.

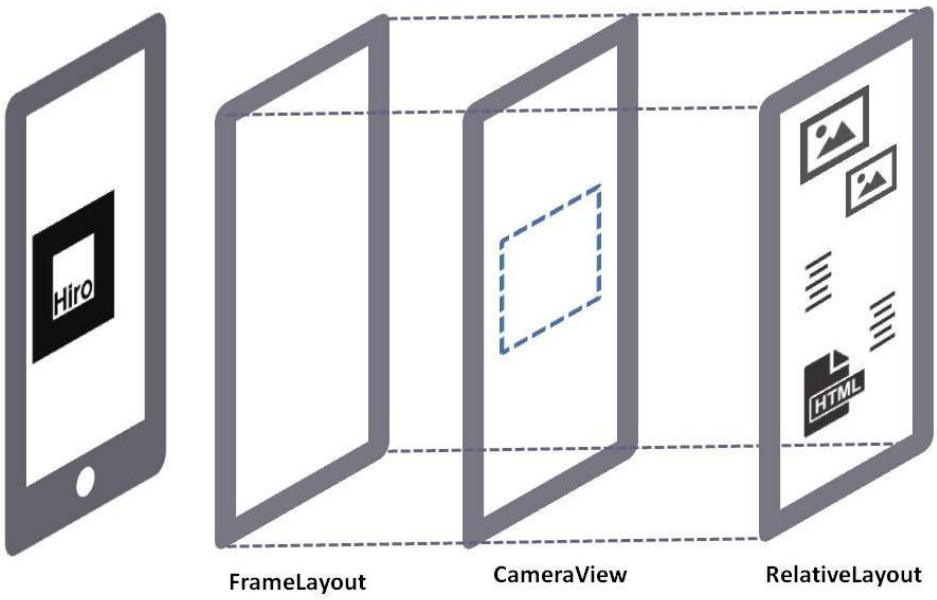


Figura 34.- Estructura de vistas del navegador de realidad aumentada

Todos estos componentes son inicializados al momento del inicio del navegador de realidad aumentada y se encuentran listos para la presentación de los objetos virtuales.

CREACIÓN DE ÁRBOL DE OBJETOS VIRTUALES

El proceso de generación de un árbol de objetos virtuales, comienza una vez se registra un marcador de realidad aumentada y se obtiene su ID (proceso descrito en la sección 4.2.8.3).

A partir de este valor, se crea un objeto de tipo *MarkerTree* que representa el árbol de objetos virtuales y contiene la información para mostrar todos los elementos en pantalla. Este árbol está compuesto por dos tipos de nodos: *MarkerNode,* que contiene la información del marcador y su *Tracker* asociado y *SceneNode* que contiene la información del objeto aumentado y provee mecanismos para que dicha información se presente.

Se han considerado tres tipos de clases que extienden la funcionalidad de la clase *SceneNode:*

- *ARTextNode:* Contiene un objeto de tipo *TextView* que puede ser presentado en la vista *RelativeLayout* del navegador. Traduce la información del objeto ARML *Text.*

- *ARImageNode:* Contiene un objeto de tipo *ImageView* que puede ser presentado en la vista *RelativeLayout* del navegador. Traduce la información del objeto ARML *Image.*

- *ARLabelNode:* Contiene un objeto de la web que será presentada en la vista *WebLayout* del navegador. Traduce la información del objeto ARML *Label.*

El proceso comienza creando la raíz del árbol *MarkerTree* mediante un nodo *MarkerNode* que contiene la información del ID del marcador registrado.

A continuación, se procede a buscar dentro del árbol de objetos ARML todos aquellos elementos de tipo *Trackable* que están relacionados al marcador en inspección. Cada elemento encontrado es después convertido a un elemento de tipo *SceneNode* y se añade al árbol.

Continuando con el caso en estudio de la Figura 28, la Figura 35 muestra a grandes rasgos cómo un árbol de objetos ARML se convierte a un árbol de objetos virtuales que se encuentra listo para ser presentado en pantalla.

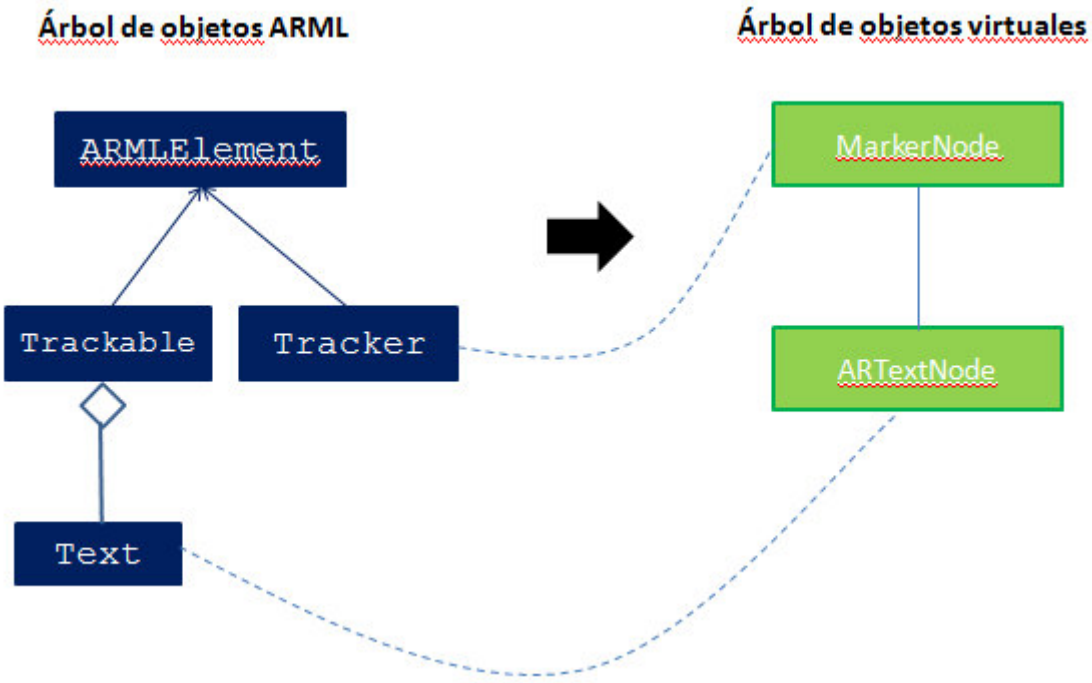


Figura 35.- Relación entre el árbol de objetos ARML y el árbol de objetos virtuales

Finalmente, la Figura 36 muestra el diagrama de secuencia de secuencia del proceso descrito, la Figura 37 muestra el diagrama de clases y las tablas Tabla 24, Tabla 25, Tabla 26, Tabla 27 y Tabla 28 dan una descripción de alto nivel de las clases más importantes.

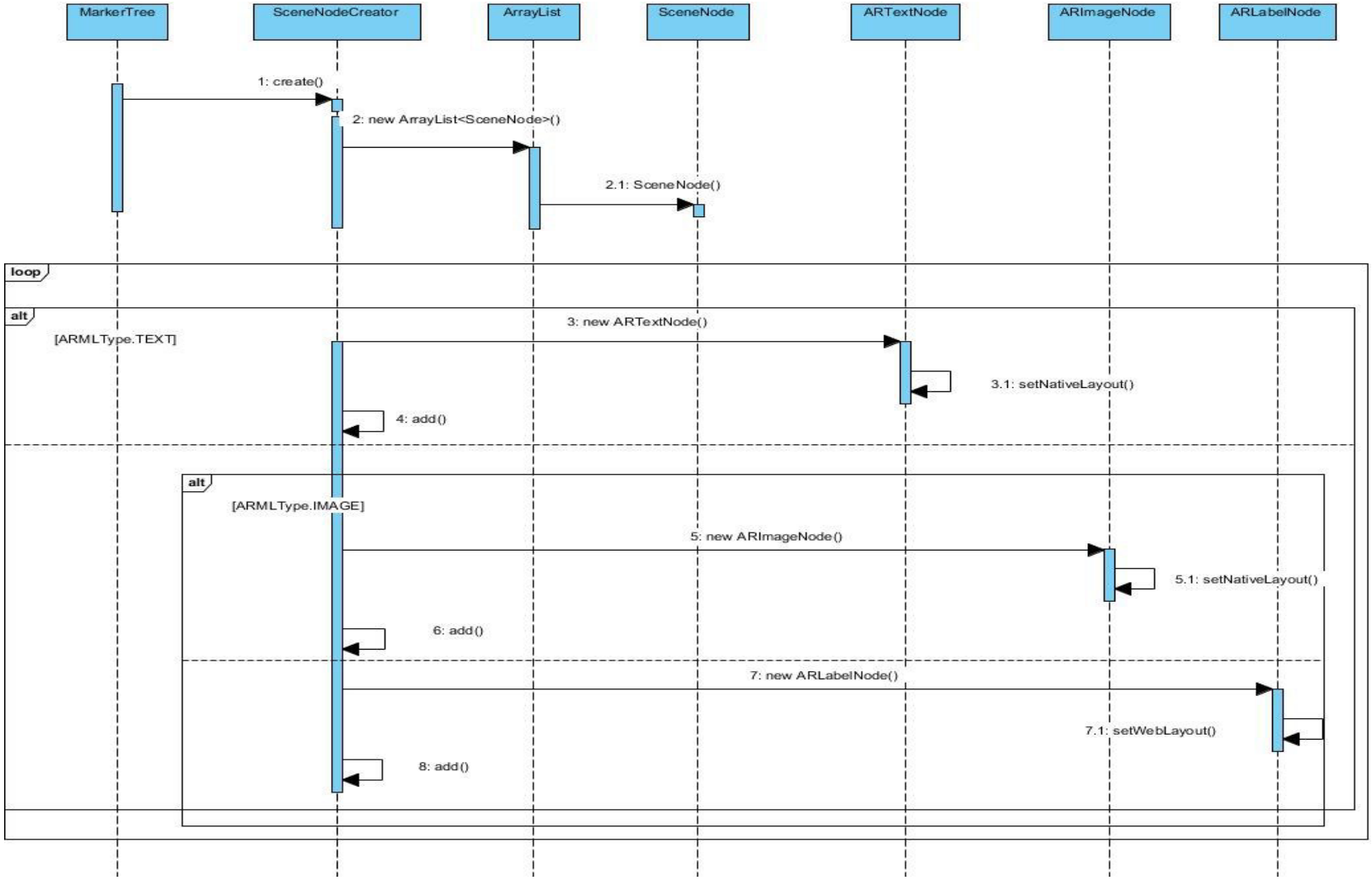


Figura 36.- Diagrama de secuencia de formación de árboles virtuales

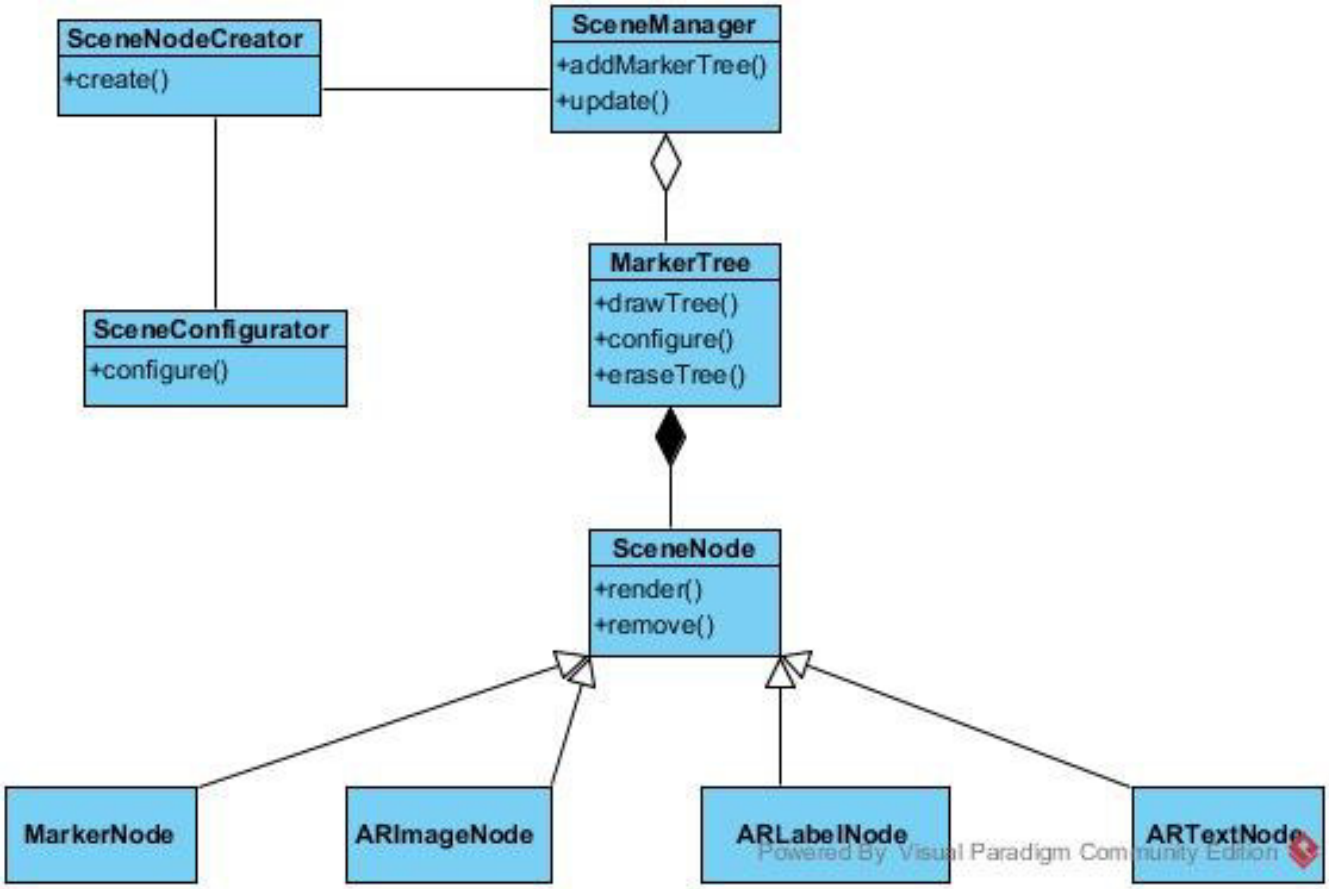


Figura 37.- Diagrama de clases del árbol de objetos virtuales

Clase	SceneConfigurator
Descripción	Realiza todas las configuraciones de las escenas de realidad aumentada
MÉTODOS	
Configure	Inicia la configuración de escenas de AR.

Tabla 24.- Descripción general clase SceneConfigurator

Clase	SceneNodeCreator
Descripción	Crea un nodo que puede ser mostrado en pantalla
MÉTODOS	
Create	Crea un nodo de tipo SceneNode

Tabla 25.- Descripción general clase SceneNodeCreator

Clase	SceneManager
Descripción	Administra el árbol de objetos virtuales
MÉTODOS	
addMarkerTree	Ingresa el árbol de objetos virtuales a administrar
Update	Actualiza el árbol de objetos virtuales según la información de la pantalla.

Tabla 26.- Descripción general SceneManager

Clase	MarkerTree
Descripción	Representa el árbol de objetos virtuales
MÉTODOS	
drawTree	Dibuja un árbol en pantalla.
configure	Configura un árbol de objetos virtuales.
eraseTree	Elimina un árbol de la pantalla.

Tabla 27.- Descripción general clase MarkerTree

Clase	SceneNode
Descripción	Un nodo genérico de un árbol de objetos virtuales
MÉTODOS	
Render	Dibuja el nodo actual en pantalla
Remove	Desdibuja el nodo actual en pantalla

Tabla 28.- Descripción general clase SceneNode

PRESENTACIÓN DE OBJETOS VIRTUALES RELACIONADOS A UN MARCADOR

La presentación del árbol de objetos virtuales es administrada por la clase *SceneManager* esta clase contiene una instancia de un árbol de objetos virtuales por cada marcador descrito y registrado en la escena de realidad aumentada ARML.

Esta clase revisa por cada *frame* de cámara si existe un marcador en pantalla asociado a uno de los árboles registrados. De existir, procede con el dibujado de cada uno de los nodos de éste árbol. Una vez que el marcador yo no se encuentra visible, se procede con el desdibujado. Este proceso se encuentra descrito en el diagrama de secuencia de la Figura 38.

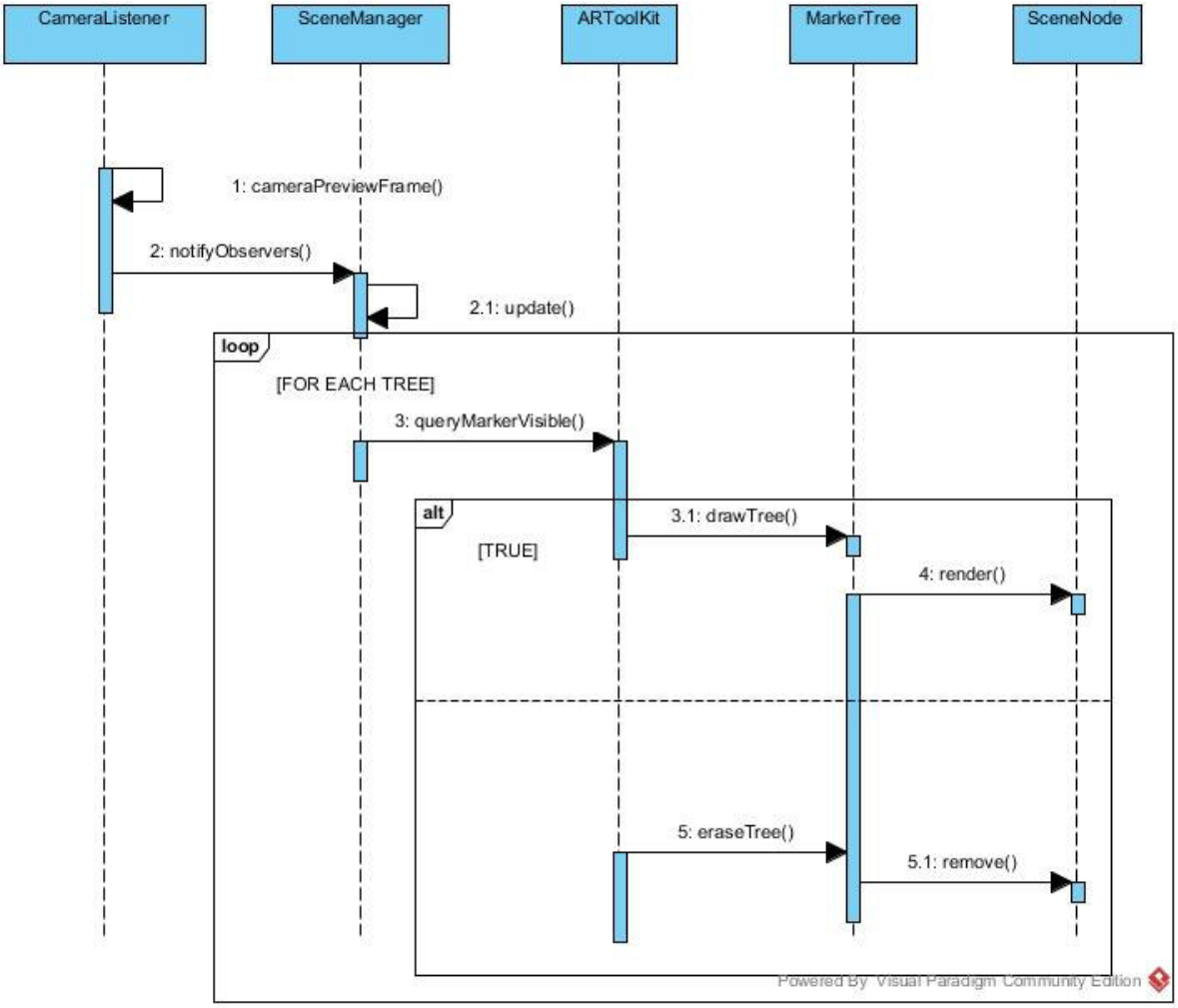


Figura 38.- Diagrama de secuencia de la presentación de objetos virtuales

4.2.9 Resultados de pruebas de la implementación

A continuación se presenta el resultado de las de la implementación descriptiva del estándar que permite validar si el navegador se comporta de acuerdo a lo definido por ARML 2.0, se utilizaron 19 pruebas que realizaban esta validación, de acuerdo a las etiquetas dentro del alcance del presente trabajo [32].

Adicionalmente se realizaron 27 escenarios de prueba que complementaban la validación realizada. El detalle de cada uno de estos escenarios se encuentra en el ANEXO IV.

.

Resultados de pruebas de conformidad sobre la implementación descriptiva			
Código	**Propósito**	**Prueba**	**Resultado**
2.1	Validar que la implementación reconoce archivos ARML2.0.	Verificar que la implementación reconoce archivos ARML2.0 correctos.	**CONFORME**
2.3	Validar que la implementación no permite ARElements con un id user.	Verificar que la implementación ignora ARElements con id = user	**CONFORME**
2.5	Validar que la implementación ignora un Anchor y sus objetos asociados cuando este es deshabilitado	Verificar que la implementación ignora cualquier Anchor y sus VissualAssets asociados cuando su propiedad enable tiene valor false.	**CONFORME**
2.6	Validar que la implementación reconoce Anchors que no tienen un Feature asociado	Verificar que la implementación añade Anchors a la escena.	**CONFORME**
2.18	Validar que la implementación no falla en caso el Tracker es desconocido.	Verificar que la implementación ignora cualquier Tracker los Trackables asociados desconocidos.	**CONFORME**
2.24	Validar que la implementación asigna correctamente el valor de orden por defecto de un TrackableConfig, en caso este no se encuentre asignado.	Verificar que la implementación asigna un valor por defecto máximo para el atributo order en caso este no haya sido definido.	**CONFORME**

2.30	Validar que la implementación ignora un VisualAsset que está deshabilitado.	Verificar que la implementación ignora un VisualAsset cuando la propiedad enable se encuentra en false.	**CONFORME**
2.31	Validar que la implementación oculta objetos correctamente.	Verificar que la implementación oculta objetos de acuerdo a la distancia indicada en el valor Zorder	**CONFORME**
2.35	Validar que la implementación cumple las reglas de precedencia en un Label.	Verificar que la implementación brinda mayor precedencia al atributo src sobre el atributo href, en caso ambos tengan valor.	**CONFORME**
2.36	Validar que la implementación no falle en caso un Label sea inválido.	Verificar que la implementación ignora Labels que tienen los parámetros src y href (ambos) no definidos.	**CONFORME**
2.39	Validar que la implementación coloca correctamente el valor por defecto para el atributo viewportWidth de un Label	Verificar que la implementación coloca el valor de 256 cuando la propiedad viewportwidth no tiene valor o tiene un valor negativo.	**CONFORME**
2.40	Validar que la implementación reemplaza correctamente los metadatos de nombre y descripción	Verificar que la implementación reemplace los metadatos $[name] y $[description] con un cadena vacía, en caso el Label no esté relacionado a un Feature	**CONFORME**
2.41	Validar que la implementación reemplaza correctamente los metadatos.	Verificar que la implementación reemplace los metadatos con formato: $[(XPath-Expression)] con un cadena vacía, en caso el Label no esté relacionado a un Feature	**CONFORME**

2.43	Validar que la implementación reemplaza correctamente los metadatos de nombre y descripción.	Verificar que la implementación reemplaza cualquier ocurrencia de $[name] y $[description] con cadena vacía en caso el elemento Text no esté relacionado a ningún label.	**CONFORME**
2.44	Validar que la implementación reemplaza correctamente los metadatos colocados en el texto.	Verificar que la implementación reemplaza cualquier ocurrencia de $[(XPath-Expression)] con cadena vacía en caso el elemento Text no esté relacionado a ningún label.	**CONFORME**
2.45	Validar que la implementación coloca el valor por defecto correcto para un elemento Text.	Verificar que la implementación coloca el valor #000000 (negro) para el parámetro color del elemento Text.	**CONFORME**
2.46	Validar que la implementación coloca el valor por defecto correcto para el color de fondo un elemento Text.	Verificar que la implementación coloca el valor #000000 (transparente) para el parámetro background-color del elemento Text.	**CONFORME**
2.47	Validar que la implementación no falle en caso un elemento Image sea inválido.	Validar que la implementación ignora un elemento Image cuando no soporta el formato de la imagen.	**CONFORME**
2.56	Validar que la ejecución de la orientación manual de VisualAsset.	Verificar que la implementación ejecuta las rotaciones en el orden correcto.	**CONFORME**

Tabla 29.- Resultados de pruebas de conformidad de la implementación descriptiva

4.3 Uso del navegador de realidad aumentada

A continuación se describe la funcionalidad del navegador de realidad aumentada implementado.

4.3.1 Descripción general de la solución

La Figura 39 muestra el funcionamiento general del navegador desarrollado. Se distinguen tres momentos importantes:

1. **Lectura de la escena basada en ARML:** En este primer momento, de acuerdo a una lógica establecida por la aplicación cliente, se lee un archivo que contiene una escena de realidad aumentada descrita con el lenguaje ARML2.0.

   A partir de esta lectura, el navegador configura todos los parámetros descritos en dicha escena identificando las relaciones existentes entre los elementos del mundo real (marcadores) y los elementos del mundo virtual (objetos aumentados).

   El navegador queda listo para escanear los elementos del mundo real.

2. **Reconocimiento de los marcadores:** En el proceso de registro del mundo real, el navegador buscará algunos de los marcadores anteriormente configurados. Cuando uno de estos objetos es encontrado, se desencadena el siguiente momento.

3. **Despliegue del objeto aumentado:** El dispositivo móvil busca los elementos del mundo virtual relacionados al objeto del mundo real encontrado y los muestra en pantalla mientras este objeto se encuentre presente.

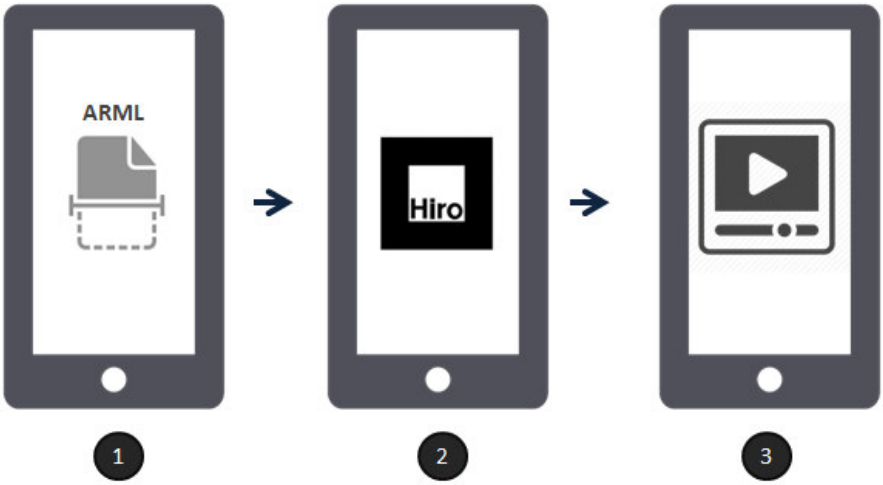


Figura 39 - Funcionamiento general del navegador de realidad aumentada

4.3.2 Interfaces del navegador de realidad aumentada

Al iniciar la aplicación, se inicia el primer proceso de realidad aumentada descrito: **Lectura del archivo ARML2.0**. En este proceso, se incluye la descarga de los objetos aumentados, la descarga de los marcadores y la configuración y registro de los marcadores de acuerdo a lo especificado en la escena de realidad aumentada. La Figura 40 muestra el desarrollo de este proceso.

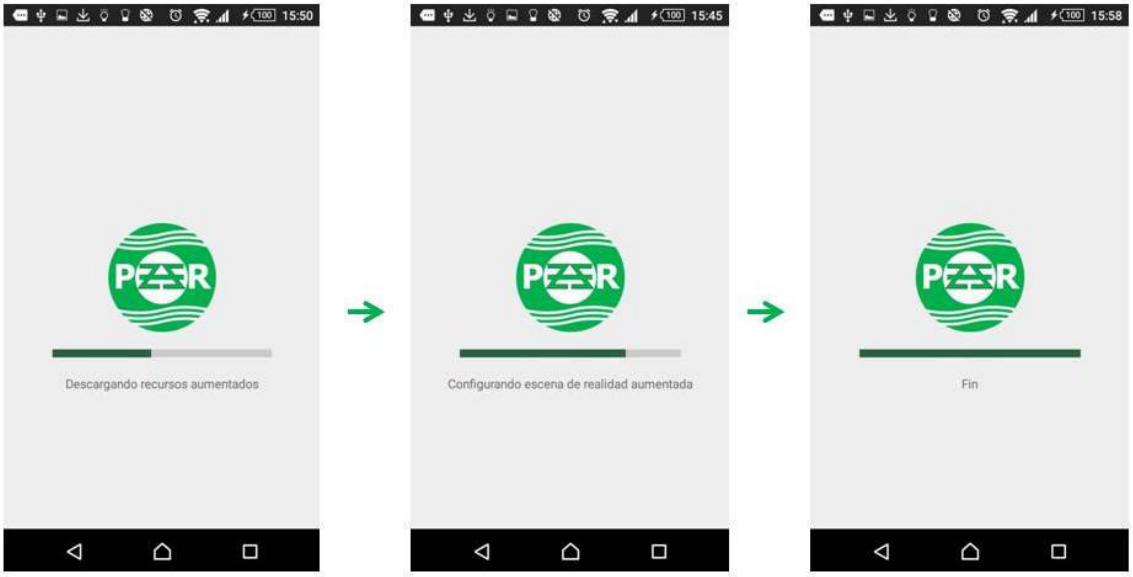


Figura 40.- Configuración de la escena de realidad aumentada

Las dos siguiente fases del proceso: **Reconocimiento de los marcadores** y **Despliegue del objeto aumentado,** se llevan a cabo una vez culminada la configuración. El proceso podrá iniciar una vez el usuario presione la pantalla y escanee un marcador. El objeto aumentado se mostrará en pantalla de acuerdo a lo especificado en el archivo ARML 2.0 mientras el marcador se encuentre visible. La Figura 41 muestra este proceso.

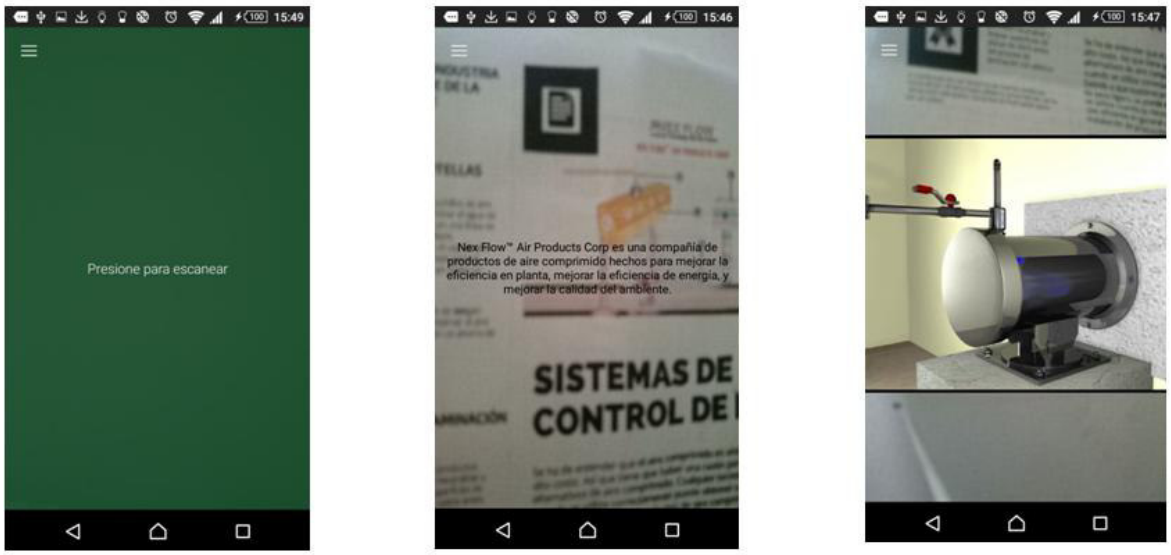


Figura 41.- Reconocimiento del marcador y despliegue de objeto aumentado

CAPÍTULO V
CASO DE ESTUDIO

En este capítulo se describe un caso de uso que muestra la aplicación del navegador de realidad aumentada para un boletín de la empresa PZZR-CAS. Comenzamos describiendo la empresa y las actividades que realiza así como sus necesidades con respecto a la tecnología de realidad aumentada. A continuación, se presenta la escena de realidad aumentada creada mediante ARML2.0 y el resultado de la interpretación de la misma por el navegador de realidad aumentada desarrollado.

5.1 PZZR-CAS

La empresa PZZR-CAS es una empresa dedicada a la consultoría y venta de tecnologías de aire comprimido para las industrias de manufactura, alimentos y bebidas, metalurgia y minería. De acuerdo a la empresa, esta tecnología le permite a las empresas reducir costos en sus operaciones y tener una gestión sustentable de su inversión industrial.

Dentro de los procesos de ventas de PZRR-CAS, se encuentra la emisión de un boletín promocional que contiene información de los productos que venden, los servicios de consultoría que brindan, casos de estudio y algunos artículos que señalan la ventajas de la tecnología que promocionan, así como noticias del sector.

La empresa decidió incorporar la tecnología de realidad aumentada en su boletín promocional, de tal forma que los recursos interactivos, tales como videos y galerías de imágenes pudieran ser referenciados directamente.

5.2 Escena de realidad aumentada y resultados

Para el presente caso de estudio, se creó una escena de realidad aumentada basada en ARML2.0. Dicha escena, contiene información sobre cinco marcadores y su relación con cinco objetos aumentados. A continuación, describimos estas relaciones.

La Tabla 30 describe la información que relaciona a un elemento aumentado de *Label* con un marcador de realidad aumentada. El marcador, se encuentra identificado mediante el elemento *Tracker* con id ***imágenes***. Este identificador permite relacionar el elemento *Label* descrito dentro de un elemento *Trackable*.

Como se ha mencionado anteriormente, un elemento *Label*, describe un documento HTML. Para este caso, dicho documento contiene una galería de imágenes animadas *GIF* que describen animaciones de los productos de la empresa PZZR-CAS.

	**Tracker ID**	imágenes
	Nombre del marcador	Imagen.patt

```
<Tracker id="imagenes">
  <uri xlink:href="http://www.pzzr-cas.com/ar/recursos/marcadores/trackers/imagen.patt"/>
</Tracker>
```

Tipo de objeto aumentado	Label
Objeto aumentado	Galería de imágenes con productos

```
<Trackable>
  <assets>
    <Label>
      <href xlink:href="http://www.pzzr-cas.com/ar/index_products.html" />
    </Label>
  </assets>
  <config >
    <tracker xlink:href="#imagenes" />
    <src>http://www.pzzr-cas.com/ar/recursos/marcadores/imagenes/imagen.JPG</src>
  </config>
</Trackable>
```

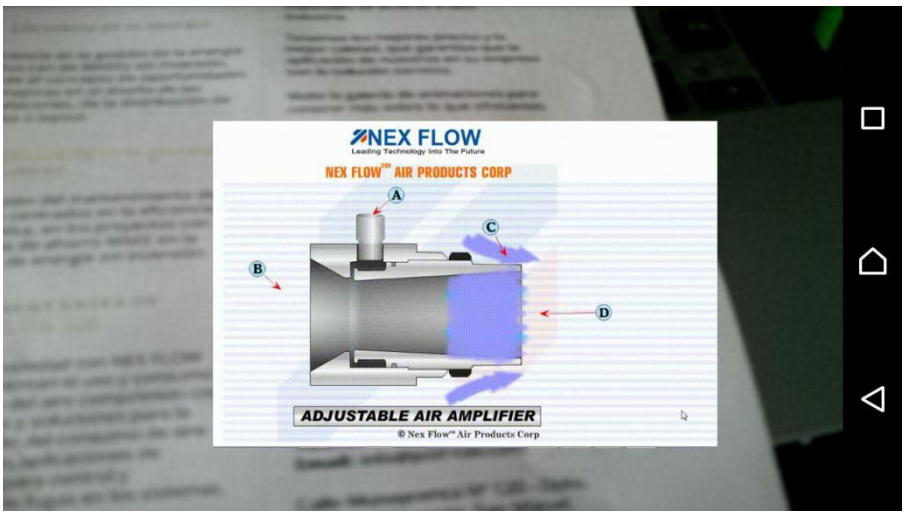


Tabla 30.- Marcador 01 del navegador de la empresa PZZR-CAS

La Tabla 31 describe la información que relaciona a un elemento aumentado de *Label* con un marcador de realidad aumentada. El marcador, se encuentra identificado mediante el elemento *Tracker* con id ***bulb***. Este identificador permite relacionar el elemento *Label* descrito dentro de un elemento *Trackable.*

En este caso, el elemento *Label*, describe un documento HTML que contiene un video con información de uno de los productos de la empresa PZZR-CAS.

	**Tracker ID**	Bulb
	Nombre del marcador	bulb.patt

```
<Tracker id="bulb">
  <uri xlink:href="http://www.pzzr-cas.com/ar/recursos/marcadores/trackers/bulb.patt"/>
</Tracker>
```

Tipo de objeto aumentado	Label
Objeto aumentado	Video

```
<Trackable>
  <assets>
    <Label>
      <href xlink:href="http://www.pzzr-cas.com/ar/index_video1.html" />
    </Label>
  </assets>
  <config >
    <tracker xlink:href="#bulb" />
    <src>http://www.pzzr-cas.com/ar/recursos/marcadores/imagenes/bulb.JPG</src
  </config>
</Trackable>
```

Tabla 31.- Marcador 02 del navegador de la empresa PZZR-CAS

La Tabla 32 también describe la información que relaciona a un elemento aumentado de *Label* con un marcador de realidad aumentada. El marcador, se encuentra identificado mediante el elemento *Tracker* con id ***quality***..

En este caso, el elemento *Label* también contiene información de un video de realidad aumentada de uno de los productos de la empresa.

	**Tracker ID**	quality
	Nombre del marcador	quality.patt

```
<Tracker id="quality">
  <uri xlink:href="www.pzzr-cas.com/ar/recursos/marcadores/trackers/quality.patt"/>
</Tracker>
```

Tipo de objeto aumentado	Label
Objeto aumentado	Video

```
<Trackable>
  <assets>
    <Label>
      <href xlink:href="http://www.pzzr-cas.com/ar/index_video2.html" />
    </Label>
  </assets>
  <config > <!-- attribute order optional -->
    <tracker xlink:href="#quality" />
    <src>http://www.pzzr-cas.com/ar/recursos/marcadores/imagenes/quality.JPG</src>
  </config>
</Trackable>
```

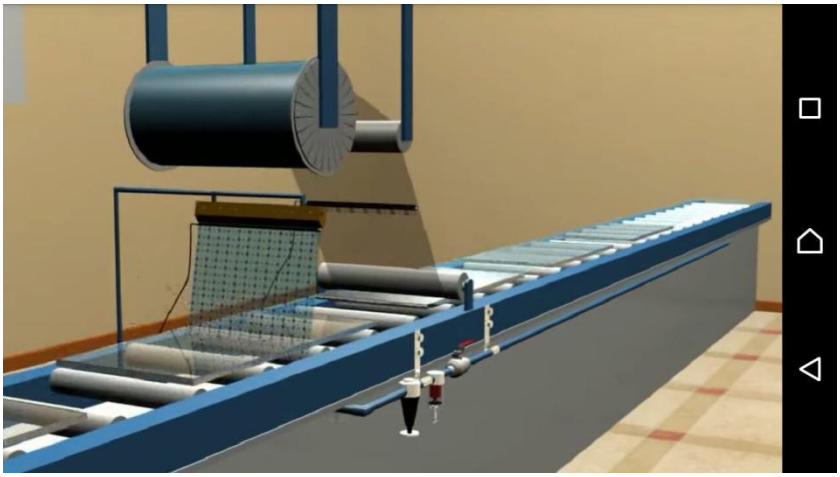


Tabla 32.- Marcador 03 del navegador de la empresa PZZR-CAS

De igual manera, la Tabla 33 presenta información que relaciona a un elemento aumentado de *Label* con un marcador de realidad aumentada. El marcador, se encuentra identificado mediante el elemento *Tracker* con id ***aire***. Este marcador al ser reconocido por el navegador de realidad aumentada despliega un video informativo de uno de los productos de la empresa PZZR-CAS.

<table>
<tr><td rowspan="2">  </td><td>Tracker ID</td><td>Aire</td></tr>
<tr><td>Nombre del marcador</td><td>aire.patt</td></tr>
<tr><td colspan="3">

```
<Tracker id="aire">
  <uri xlink:href="http://www.pzzr-cas.com/ar/recursos/marcadores/trackers/aire.patt"/>
</Tracker>
```

</td></tr>
<tr><td colspan="2">Tipo de objeto aumentado</td><td>Label</td></tr>
<tr><td colspan="2">Objeto aumentado</td><td>Video</td></tr>
<tr><td colspan="3">

```
<Trackable>
  <assets>
    <Label>
      <href xlink:href="http://www.pzzr-cas.com/ar/index_video3.html" />
    </Label>
  </assets>
  <config >
    <tracker xlink:href="#aire" />
    <src>http://www.pzzr-cas.com/ar/recursos/marcadores/imagenes/aire.JPG</src
  </config>
</Trackable>
```

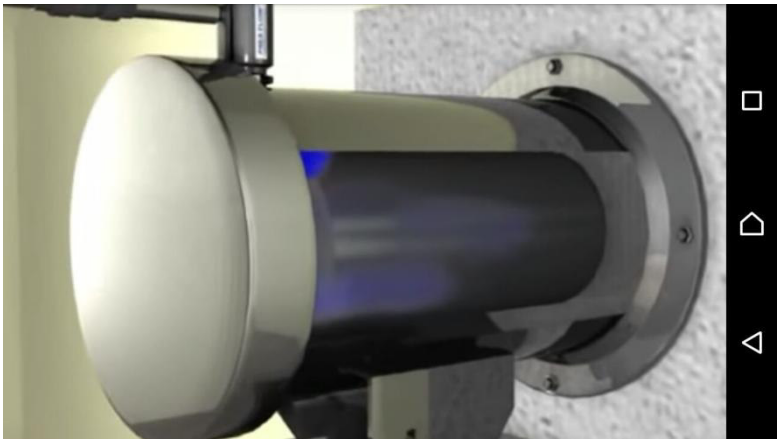
</td></tr>
<tr><td colspan="3">  </td></tr>
</table>

Tabla 33.- Marcador 04 del navegador de la empresa PZZR-CAS

Finalmente, la Tabla 34 presenta información que relaciona a un elemento aumentado de *Text* con un marcador de realidad aumentada. El marcador, se encuentra identificado mediante el elemento *Tracker* con id ***texto***.

Como resultado de esta relación, el navegador de realidad aumentada despliega un texto definido al reconocer el marcador de realidad aumentada asociado.

	**Tracker ID**	Texto
	Nombre del marcador	text.patt

```
<Tracker id="texto">
  <uri xlink:href="http://www.pzzr-cas.com/ar/recursos/marcadores/trackers/text.patt"/>
</Tracker>
```

Tipo de objeto aumentado	Text
Objeto aumentado	Texto

```
<Trackable>
  <assets>
    <Text>
      <src>Nex Flow™ Air Products Corp es una compañía de productos de aire comprimido hechos
        para mejorar la eficiencia en planta, mejorar la eficiencia de energía, y mejorar la calidad del ambiente.</src
    </Text>
  </assets>
  <config > <!-- attribute order optional -->
    <tracker xlink:href="#texto" />
    <src>http://www.pzzr-cas.com/ar/recursos/marcadores/imagenes/texto.JPG</src>
  </config>
</Trackable>
```

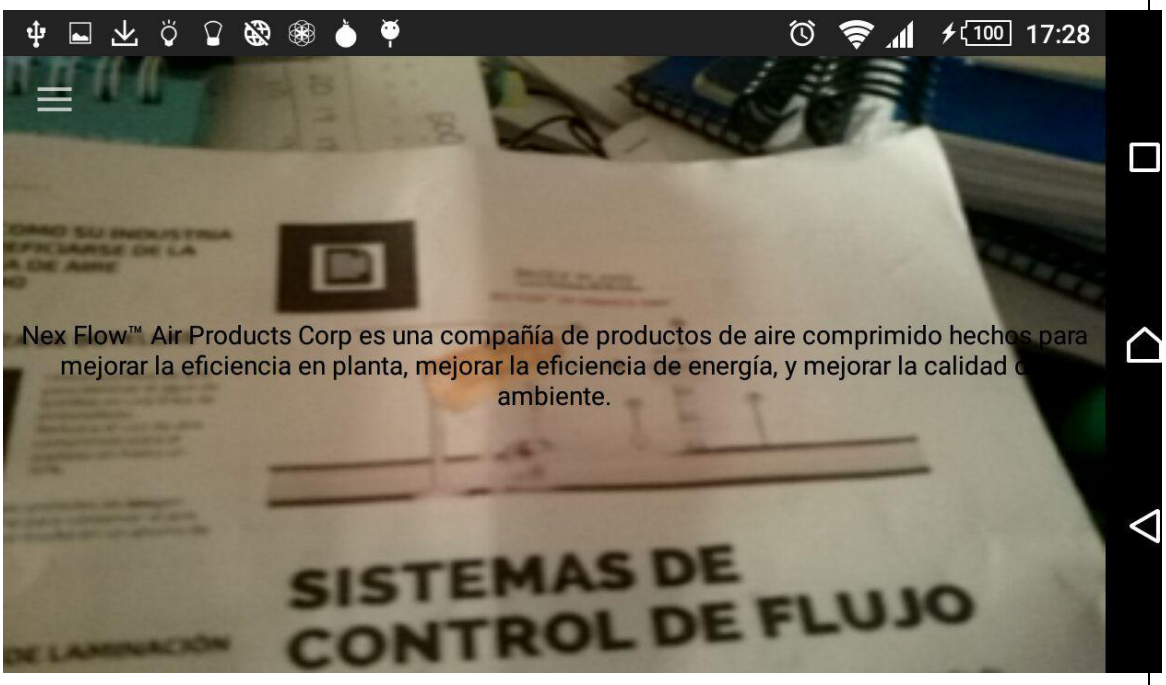


Tabla 34.- Marcador 05 del navegador de la empresa PZZR-CAS

CAPÍTULO VI
CONCLUSIONES Y TRABAJOS FUTUROS

6.1 Conclusión General

El presente trabajo de investigación tiene como objetivo general desarrollar un navegador de realidad aumentada para aplicaciones basadas en la visión, que reconozca marcadores mediante el estándar ARML 2.0.

Dicho objetivo se ha cumplido al presentar la implementación del navegador de realidad aumentada en el capítulo 4, también se presentaron los resultados de las pruebas que validan que la implementación realizada cumple con el estándar ARML 2.0.

Adicionalmente, se ha realizado un caso de estudio descrito en el capítulo 5, donde se puede ver cómo una escena genérica descrita mediante ARML 2.0 puede ser reconocida por el navegador.

6.2 Conclusiones Específicas

6.2.1 Objetivo Específico 1

Para el cumplimiento del objetivo específico 1, se analizaron diferentes *frameworks*, navegadores y plataformas de realidad aumentada. Dicho estudio se realizó en el capítulo 3 donde se describieron los navegadores MARW [42], ARGON [43], Layar[44], Wikitude[16] y Vuforia[47].

6.2.2 Objetivo Específico 2

Para el cumplimiento del objetivo específico 2, se estudió el sistema operativo Android en el capítulo 2 y se desarrolló el navegador de realidad aumentada sobre esta plataforma en el capítulo 4.

6.2.3 Objetivo Específico 3

Para el cumplimiento del objetivo específico 3, se estudió el concepto de marcadores de realidad aumentada en el capítulo 2, y se analizaron diferentes *frameworks* que permiten el reconocimiento de marcadores, tales como ARToolKit [29], Alvar[14], BazAR[53] y DroidAR[54] en el capítulo 3. Finalmente, se desarrolló un navegador de realidad aumentada capaz de reconocer marcadores (capítulo 4), y se aplicó el uso del navegador a un caso de estudio en el capítulo 5.

6.2.4 Objetivo Específico 4

Para el cumplimiento del objetivo específico 4, se realizó un estudio del lenguaje ARML 2.0 en el capítulo 2 y se tomó como base la librería ARToolKit para el desarrollo del navegador de realidad aumentada descrito en el capítulo 4.

6.3 Trabajos Futuros

Los trabajos futuros que se pueden considerar son los siguientes:

- Ampliar la funcionalidad del navegador de realidad aumentada implementando otras partes del estándar ARML2.0 que permitan crear aplicaciones basadas en la localización.

- Ampliar la funcionalidad del navegador de realidad aumentada implementando la parte *script* del lenguaje ARML 2.0.

- Ampliar la funcionalidad del navegador de realidad aumentada implementando el lenguaje COLLADA para el reconocimiento de figuras 3D.

- Crear de una plataforma de realidad aumentada basada en ARML2.0, dicha plataforma podría incluir una herramienta para la publicación de contenido aumentado.

REFERENCIAS BIBLIOGRÁFICAS

[1] C. Arth, R. Grasset, L. Gruber, T. Langlotz, A. Mulloni, and D. Wagner, "The History of Mobile Augmented Reality," *arXiv Prepr. arXiv1505.01319*, 2015.

[2] T. P. Caudell and D. W. Mizell, "Augmented reality: an application of heads-up display technology to manual manufacturing processes," in *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*, 1992, vol. ii, pp. 659–669 vol.2.

[3] J. P. Rolland, D. L. Wright, and A. R. Kancherla, "Towards a novel augmented-reality tool to visualize dynamic 3-D anatomy," *Stud. Health Technol. Inform.*, pp. 337–348, 1997.

[4] T. Riess and S. Weghorst, "Augmented reality in the treatment of Parkinson's disease," *Proc. Med. Meets Virtual Real. III, San Diego*, 1995.

[5] S. Lavallée, P. Cinquin, R. Szeliski, O. Peria, A. Hamadeh, G. Champleboux, and J. Troccaz, "Building a hybrid patient's model for augmented reality in surgery: A registration problem," *Comput. Biol. Med.*, vol. 25, no. 2, pp. 149–164, 1995.

[6] W. E. Mackay and A. Fayard, "Designing Interactive Paper: Lessons from three Augmented Reality Projects," in *In Proceedings of IWAR'98, International Workshop on Augmented Reality*, 1999, pp. 81–90.

[7] D. Sims, "New realities in aircraft design and manufacture," *Comput. Graph. Appl. IEEE*, vol. 14, no. 2, p. 91-, 1994.

[8] N. R. Corby Jr. and C. A. Nafis, "Augmented reality telemanipulation system for nuclear reactor inspection," *Proc. SPIE*, vol. 2351. pp. 360–365, 1995.

[9] W. A. Hoff, K. Nguyen, and T. Lyon, "Computer-vision-based registration techniques for augmented reality," *Proc. SPIE*, vol. 2904. pp. 538–548, 1996.

[10] S. Ravela, "Tracking object motion across aspect changes for augmented reality," *Comput. Sci. Dep. Fac. Publ. Ser.*, p. 220, 1996.

[11] J. M. Levin, "Real-time target and pose recognition for 3-d graphical overlay," Massachusetts Institute of Technology, 1997.

[12] S. Feiner, B. MacIntyre, T. Höllerer, and A. Webster, "A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment," *Pers. Technol.*, vol. 1, no. 4, pp. 208–217, 1997.

[13] H. Kato, "ARToolKit," *http://www. hitl. washington. edu/artoolkit/*, 1999.

[14] VTT Technical Research Centre of Finland, "Augmented Reality / 3D Tracking | ALVAR," 2012. [Online]. Available: http://virtual.vtt.fi/virtual/proj2/multimedia/alvar/index.html. [Accessed: 01-Feb-2016].

[15] A. Hill, B. MacIntyre, M. Gandy, B. Davidson, and H. Rouzati, "Kharma: An open kml/html architecture for mobile augmented reality applications," in *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*, 2010, pp. 233–234.

[16] "Wikitude - Android Native - Devzone." [Online]. Available:

http://www.wikitude.com/developer/documentation/androidnative. [Accessed: 28-Jan-2016].

[17] "Augmented Reality | Interactive Print | Layar." [Online]. Available: https://www.layar.com/. [Accessed: 27-Jan-2016].

[18] S. Ahn, H. Ko, and B. Yoo, "Webizing mobile augmented reality content," *New Rev. Hypermedia Multimed.*, vol. 20, no. 1, pp. 79–100, 2014.

[19] B. MacIntyre, H. Rouzati, and M. Lechner, "Walled Gardens: Apps and Data as Barriers to Augmenting Reality," *IEEE Comput. Graph. Appl.*, vol. 33, no. 3, pp. 77–81, May 2013.

[20] M. Lechner, "A proposed step-by-step guide to an AR standard," *Wikitude GmbH*, 2011.

[21] "OGC® adopts ARML 2.0, new open standard encoding for sharing Augmented Reality data | OGC." [Online]. Available: http://www.opengeospatial.org/pressroom/pressreleases/2180.

[22] "ARML 2.0 SWG | OGC(R)." [Online]. Available: http://www.opengeospatial.org/projects/groups/arml2.0swg. [Accessed: 26-Jan-2016].

[23] "AR Browser Interoperability Proof of Concept – AR Community." [Online]. Available: http://www.perey.com/ARStandards/ar-browser-interoperability-proof-of-concept/. [Accessed: 26-Apr-2016].

[24] "Gartner IT Glossary - Augmented Reality (AR)." [Online]. Available: http://www.gartner.com/it-glossary/augmented-reality-ar/. [Accessed: 15-May-2013].

[25] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino, "Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum," 1994, pp. 282–292.

[26] V. Geroimenko, "Augmented Reality Technology and Art: The Analysis and Visualization of Evolving Conceptual Models," in *Information Visualisation (IV), 2012 16th International Conference on*, 2012, pp. 445–453.

[27] J. Jung, J. Ha, S.-W. Lee, F. A. Rojas, and H. S. Yang, "Efficient mobile AR technology using scalable recognition and tracking based on server-client model," *Comput. Graph.*, vol. 36, no. 3, pp. 131–139, May 2012.

[28] L. Madden, *Professional Augmented Reality Browsers for Smartphones: Programming for junaio, Layar and Wikitude*, 1st ed. Wiley Publishing, 2011.

[29] "Open Source Augmented Reality SDK | ARToolKit.org." [Online]. Available: http://artoolkit.org/. [Accessed: 27-Jan-2016].

[30] T. Reicher, "A framework for dynamically adaptable augmented reality systems.," Technical University Munich, 2004.

[31] B. Butchart, "Architectural styles for augmented reality in smartphones," in *Third International AR Standards Meeting*, 2011, pp. 1–7.

[32] Open Geospatial Consortium, "OGC® Augmented Reality Markup Language 2.0 (ARML 2.0)." Open Geospatial Consortium, 24-Feb-2015.

[33] D. Rumi'nski and K. Walczak, "CARL: a language for modelling contextual augmented reality environments," in *Technological innovation for collective awareness systems*, Springer, 2014, pp. 183–190.

[34] “Android - Discover Android.” [Online]. Available: http://www.android.com/about/. [Accessed: 06-Jun-2013].

[35] C. Collins, M. Galpin, and M. Kaeppler, *Android in Practice*, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2011.

[36] M. Lettner, M. Tschernuth, and R. Mayrhofer, “Mobile platform architecture review: android, iphone, qt,” in *Proceedings of the 13th international conference on Computer Aided Systems Theory - Volume Part II*, 2012, pp. 544–551.

[37] “Glossary | Android Developers.” [Online]. Available: http://developer.android.com/guide/appendix/glossary.html. [Accessed: 07-Jun-2013].

[38] “Pausing and Resuming an Activity | Android Developers.” [Online]. Available: http://developer.android.com/training/basics/activity-lifecycle/pausing.html. [Accessed: 07-Jun-2013].

[39] “Managing the Activity Lifecycle | Android Developers.” [Online]. Available: http://developer.android.com/training/basics/activity-lifecycle/index.html. [Accessed: 09-Jun-2013].

[40] “Starting an Activity | Android Developers.” [Online]. Available: http://developer.android.com/training/basics/activity-lifecycle/starting.html. [Accessed: 07-Jun-2013].

[41] “Stopping and Restarting an Activity | Android Developers.” [Online]. Available: http://developer.android.com/training/basics/activity-lifecycle/stopping.html. [Accessed: 07-Jun-2013].

[42] A. Karhu, A. Heikkinen, and T. Koskela, “Towards Augmented Reality Applications in a Mobile Web Context,” in *Next Generation Mobile Apps, Services and Technologies (NGMAST), 2014 Eighth International Conference on*, 2014, pp. 1–6.

[43] B. MacIntyre, A. Hill, H. Rouzati, M. Gandy, and B. Davidson, “The Argon AR Web Browser and standards-based AR application environment,” in *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, 2011, pp. 65–74.

[44] “What is Layar? – Layar.” [Online]. Available: http://www.layar.com/what-is-layar/. [Accessed: 13-Jun-2013].

[45] “Layar Platform Overview – Layar Developer Documentation.” [Online]. Available: http://www.layar.com/documentation/browser/layar-platform-overview/. [Accessed: 15-Jun-2013].

[46] “Wikitude SDK Android Native API 1.2.0 Documentation.” [Online]. Available: http://www.wikitude.com/external/doc/documentation/latest/androidnative/gettingstartedcloudrecognition.html#getting-started-with-the-cloud-recognition-service. [Accessed: 28-Jan-2016].

[47] “Getting Started View | Vuforia Library Prod.” [Online]. Available: http://developer.vuforia.com/library/getting-started. [Accessed: 29-Jan-2016].

[48] “Vuforia SDK Architecture | Vuforia Developer Portal.” [Online]. Available: https://developer.vuforia.com/resources/dev-guide/vuforia-ar-architecture. [Accessed: 22-Jun-2013].

[49] D. Amin and S. Govilkar, “COMPARATIVE STUDY OF AUGMENTED

REALITY SDK'S," *Int. J. Comput. Sci. Appl.*, vol. 5, no. 1, 2015.

[50] "Download the ARToolKit Augmented Reality SDK | ARToolKit.org." [Online]. Available: http://artoolkit.org/download-artoolkit-sdk. [Accessed: 27-Jan-2016].

[51] "About ARToolKit's History and Team | ARToolKit.org." [Online]. Available: http://artoolkit.org/about-artoolkit. [Accessed: 27-Jan-2016].

[52] "Developing with ARToolKitWrapper and ARBaseLib [ARToolkit]." [Online]. Available: http://artoolkit.org/documentation/doku.php?id=4_Android:android_developing. [Accessed: 28-Jan-2016].

[53] École Polytechnique Federale de Laussane Computer Vision Laboratory CVLAB, "BazAR: A vision based fast detection library | CVLAB." [Online]. Available: http://cvlab.epfl.ch/software/bazar. [Accessed: 01-Feb-2016].

[54] Z. Huang, P. Hui, C. Peylo, and D. Chatzopoulos, "Mobile augmented reality survey: a bottom-up approach," *CoRR*, vol. abs/1309.4, 2013.

[55] D. Jooste, V. Rautenbach, and S. Coetzee, "RESULTS OF AN EVALUATION OF AUGMENTED REALITY MOBILE DEVELOPMENT FRAMEWORKS FOR ADDRESSES IN AUGMENTED REALITY," in *Free and Open Source Software for Geospatial*, 2015.

[56] "Learn about sprint artifacts and spring activities - Scrum Alliance." [Online]. Available: http://www.scrumalliance.org/why-scrum/core-scrum-artifacts-activities. [Accessed: 17-Sep-2013].

[57] S. B. Kaleel and S. Harishankar, *Applying Agile Methodology in Mobile Software Engineering: Android Application Development and its Challenges*. 2013.

[58] K. S. Rubin, *Essential Scrum: A practical guide to the most popular Agile process*. Addison-Wesley, 2012.

[59] "Scrum Guide | Scrum Guides." [Online]. Available: http://www.scrumguides.org/scrum-guide.html. [Accessed: 21-Jan-2016].

[60] D. C. S. R. Greg Lavender, "Active Object -- An Object Behavioral Pattern for Concurrent Programming."

[61] A. Goransoon, *Efficient Android Threading*. O'Reilly Media, Inc., 2014.

[62] "W3C Document Object Model." [Online]. Available: https://www.w3.org/DOM/. [Accessed: 03-May-2016].

[63] A. Visser, "Survey of XML Languages for Augmented Reality Content," in *Proc. of AR Standardization Forum, Barcelona*, 2011.

[64] "KARML Reference | KHARMA." [Online]. Available: https://research.cc.gatech.edu/kharma/content/karml-reference. [Accessed: 29-Jan-2016].

ANEXOS

ANEXO I: DOCUMENTACIÓN DEL ESTÁNDAR ARML 2.0

A continuación se presenta una descripción de todos los elementos que componen el lenguaje ARML 2.0

Elemento	Tipo de Realidad Aumentada	Descripción
ARElement	Visión. Geolocalización	Elemento inicial de un archivo ARLM 2.0.
Feature	Visión. Geolocalización	Elemento que permite agrupar objetos. Para aplicaciones basadas en la geolocalización representa el lugar físico del mundo real. En las aplicaciones basadas en la visión, representa un texto que describe la relación marcador - objeto aumentado.
Anchor	Visión. Geolocalización	Elemento genérico usado por el navegado de realidad aumentada. Puede ser ARAnchor o ScreenAnchor.
ARAnchor	Visión	Representa un objeto del mundo real.
Geometry	Geolocalización	Permite describir figuras geométricas.
GMLGeometries	Geolocalización	Permite describir figuras geométricas mediante el estándar GLM.
Point	Geolocalización	Permite describir un punto mediante el estándar GLM.
LineString	Geolocalización	Permite describir una línea mediante el estándar GLM.
Polygon	Geolocalización	Permite describir un polígono mediante el estándar GLM.
RelativeTo	Geolocalización	Permite relacionar un elemento aumentado con respecto a otro.
Tracker	Visión	Permite representar la un marcador de realidad aumentada.

TrackableConfig	Visión	Permite relacionar un marcador con un grupo de objetos aumentados.
Trackable	Visión	Permite describir un grupo de elementos aumentados, del mundo virtual.
VisualAsset	Visión	Permite representar objetos aumentados 2D o 3D.
ScreenAnchor	--	Permite describir una zona del navegador con contenido fijo. Ej: En juegos de realidad aumentada esta zona podría mostrar el puntaje del jugador o avisos de la aplicación.
VisualAsset2D	Visión	Interfaz para aumentar objetos 2D tales como texto, imágenes, figuras o contenido HTML.
Label	Visión	Permite aumentar contenido descrito en HTML.
Text	Visión	Permite aumentar texto.
Image	Visión	Permite aumentar imágenes 2D.
Fill	Visión	Permite aumentar figuras geométricas. Requiere la implementación del estándar GLM.
Model	Visión	Permite aumentar objetos 3D. ARML2.0 sugiere la implementación e integración con el estándar COLLADA .
Scale	Visión. Geolocalización	Permite describir la escala de los recursos aumentados.
Condition	Visión. Geolocalización	Permite describir condiciones de distancia o de selección bajo las cuales se muestran los recursos aumentados. Requiere de la parte *script* de ARML 2.0

Tabla 35.- Elementos del lenguaje ARML2.0

ANEXO II: LENGUAJES PARA DESCRIBIR ESCENAS DE REALIDAD AUMENTADA

Un lenguaje de realidad aumentada, permite describir una escena donde se identifican los objetos del mundo real, los objetos del mundo virtual y las relaciones existentes entre ellos. En este subcapítulo se presentan algunos lenguajes creados en otros trabajos.

CARL

El lenguaje CARL - *Contextual Augmented Reality Language*, es un lenguaje de realidad aumentada orientado para aplicaciones basadas en la geolocalización y en la visión sin el uso de marcadores (*markerless augmented reality*). Los autores proponen un ecosistema de realidad aumentada basada en el contexto, donde las escenas son dinámicas y toman en consideración elementos como la ubicación, las preferencias de usuario o las características del dispositivo; dicho ecosistema debería permitir el uso de múltiples fuentes de datos y el acceso a diferentes servicios de realidad aumentada combinando toda la información obtenida en una sola escena de realidad aumentada [33].

El lenguaje CARL define tres tipos de entidades:

- **Trackables:** Etiquetas que son usadas para describir objetos del mundo real. Estos objetos son identificados de manera única mediantes URIs.

  Una etiqueta *Trackable* tiene tres etiquetas hijas que definen tres momentos de ejecución:

  - **Begin:** Define un conjunto de instrucciones a ejecutar cuando el elemento *Trackable* es reconocido.

  - **Active:** Define un conjunto de instrucciones a ejecutar cuando el elemento *Trackable* es visible.

  - **End:** Define un conjunto de instrucciones a ejecutar cuando el elemento *Trackable* desaparece del rango de visión.

- **Content Objects:** Esta etiqueta contiene los recursos multimedia que son parte del mundo virtual. Es decir, contiene el objeto aumentado. Cada *ContentObject* tiene etiquetas hijas llamadas *Resources* que permiten la definición recursos como componentes y localización de los objetos multimedia y *Actions*, que permiten definir un conjunto de acciones a ejecutar sobre estos recursos.

- **Interfaces:** Esta etiqueta permite definir las interacciones entre el usuario y el dispositivo que se está utilizando.

KARML

KARML es un lenguaje que fue creado a partir del estándar KML. Permite describir escenas para aplicaciones de realidad aumentada basadas en la visión y en la geolocalización.

Una de las principales ventajas de este lenguaje es que permite añadir opciones de estilo mediante CSS e interacción con los elementos de una escena mediante JavaScript; también permite definir la ubicación de ciertos elementos en relación a otros [63].

Para el caso de aplicaciones basadas en la visión, KARML define un elemento denominado ***KMLTracker*** que permite definir características del marcador a utilizar como un ID, la medida en metros del mismo y la orientación.

Un ejemplo de una escena que permite el reconocimiento de marcadores utilizando KARML, puede verse a continuación en la Figura 42 [64]:

```
<Placemark>
        <name>Tracker Fiducial Marker Example</name>
        <Tracker>
                <device>#simpleid</device>
                <options>
                        <markerId>31</markerId>
                        <width>0.07</width>
                        <height>0.07</height>
                </options>
                <locationMode>relative</locationMode>
                <orientationMode>billboard</orientationMode>
        </Tracker>
</Placemark>
```

Figura 42 - Descripción de una escena de realidad aumentada usando KARML

La Tabla 36 explica cada uno de los campos descritos:

Campo	Descripción
Placemark	Esta es una etiqueta tomada de KML. Permite definir características sobre un punto específico.
Name	El nombre brindado al Placemark
Tracker	Etiqueta añadida por KARML. Permite relacionar un marcador a la escena de realidad aumentada.
Device	Describe el marcador a utilizar.
Options	Permite describir opciones y características relacionadas al marcador.
MarkerId	Definer un número asignado al marcador. Si se coloca -1, la aplicación basada en KARML buscará cualquier marcador visible.
Width	El ancho del marcador en metros.
Height	El largo del marcador en metros. Opcional
LocationMode	Permite describir el modo de localización del marcador con respecto a otro.
OrientationMode	Permite describir la orientación del marcador con respecto a otro.

Tabla 36- Descripción de una escena de realidad aumentada usando KARML

La descripción de los objetos aumentados puede realizarse utilizando las etiquetas propias de KML (dado que KARML es sólo una extensión de este lenguaje).

HTML

Algunos trabajos como [18] proponen que las aplicaciones de realidad aumentada deben basarse en los estándares de la Web. Este enfoque permitiría utilizar los lenguajes HTML, CSS y JavaScript para modelar escenas de realidad aumentada.

El enfoque descrito en [18], propone "hacer uso de la estructura existente del estándar HTML extendiendo el uso de URIs para identificar y referenciar objetos físicos y lugares". Los autores indican que la "Web tiene el potencial de relacionar el mundo real al mundo virtual" y que " en el modelo de información de la Web, un recurso es una primitiva que un hipervínculo puede alcanzar, y el hipervínculo está dirigido por una URI. Una URI no contiene ninguna información cuando es referenciada pero provee representaciones como respuesta a un navegador Web."

Los autores utilizan CSS y una propiedad a la que denominaron -ar-target para colocar las URIs que representan los objetos del mundo real. A continuación se muestra uno de los ejemplos indicados en su trabajo [18]:

```
<html>
  <head>
    <style type="text/css" media="place">
      .media {
        -ar-target: object ("http://example.org/poi/21");
      }
    </style>
  </head>
  <body>
    <img class="media" src="gift_animation.gift" />
    <video class="media" src="html5_video.ogg" controls />
    <svg  class="media" xmlns="http://www.w3.org/2000/svg" version="1.1">
      <g>
        <polygon points="0,0 100,0 100,0 0,100" style="fill:line">
      </g>
    </svg>
  </body>
</html>
```

El ejemplo anterior nos permite definir una escena de realidad aumentada donde un objeto del mundo real, que se encuentra definido en el elemento -ar-target mediante hojas de estilo CSS es aumentado por tres objetos virtuales (una imagen, un video y una imagen vectorial) definidos dentro de la etiqueta body del documento HTML.

Comparación de lenguajes de realidad aumentada

Basándonos en los estudios realizados en [18] y [63] se han definido 7 indicadores que nos permiten comparar los diferentes lenguajes de realidad aumentada estudiados.

- **ARTYPE - Tipo de realidad aumentada:** Este indicador nos ayuda a validar qué tipos de realidad aumentada son soportados por cada uno de los lenguajes estudiados. Definimos los siguientes tipos:

    - **G:** Realidad aumentada basada en la geolocalización.
    - **M:** Realidad aumentada basada en marcadores.
    - **N:** Realidad aumentada basada en la visión sin marcadores (*markerless*)

- **VRTUAL - Tipo de objetos virtuales:** Este indicador nos permite identificar qué tipos de objetos virtuales son soportados por cada lenguaje.

- **STRUCT - Estructura del documento:** Este indicador está orientado a definir la tecnología base sobre la cual se basa el documento.

- **VISUAL - Control sobre visualización:** Este indicador nos ayuda a definir las maneras en las que el lenguaje nos da control sobre forma en la que se visualizan los objetos de realidad aumentada. Se incluyen parámetros como la escala, la orientación y otros como color y forma . Definimos los siguientes tipos:

    - **E:** El control se da mediante etiquetas definidas en el lenguaje.
    - **C:** El control mediante CSS.

- **SCRIPT - Componente de lenguaje script:** Este indicador está orientado a definir si el lenguaje provee alguna especificación para componentes basados en

scripts que permita modelar las interacciones de una aplicación de realidad aumentada.

- **TESTAR - Definición de pruebas:** Este indicador nos permite visualizar el conjunto de pruebas que ofrece el lenguaje para validar que un archivo basado en el mismo sea correcto.

- **STANDR - Estándares:** Este indicador nos permite validar si el lenguaje propuesto constituye un estándar o no.

Lenguaje	ARTYPE			VRTUAL	STRUCT	VISUAL		SCRIPT	TESTAR	STANDR
	G	M	N			E	C			
CARL	X		X	Imágenes 2D Objetos 3D	XML	X				
KARML	X	X	X	Imágenes 2D Objetos 3D	KML	X	X	X		
ARML 2.0	X	X	X	HTML Texto Imagen Objetos 3D	XML	X	X	X	X	X
Basado en HTML	X	X	X	Estándar HTML: Video Imagen Hipervínculos Texto	HTML		X	X		

Tabla 37 - Comparación de lenguajes de realidad aumentada

ANEXO III: HISTORIAS DE USUARIO

A continuación, se presentan las historias de usuario creadas.

Historia de Usuario		
Número: 1	**Usuario:** Cliente	**Prioridad:** Alta
Título: Reconocer escenas basadas en ARML2.0		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada reconozca escenas basadas en el estándar ARML2.0		
Observaciones:		

Historia de Usuario		
Número: 2	**Usuario:** Cliente	**Prioridad:** Alta
Título: Obtener un archivo ARML2.0 desde una URL		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada lea un archivo ARML 2.0 desde una URL.		
Observaciones:		

Historia de Usuario		
Número: 3	**Usuario:** Cliente	**Prioridad:** Alta
Título: Aumentar objetos tipo Texto		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada pueda mostrar elementos aumentados de tipo Texto		
Observaciones: El texto es leído desde el atributo src.		

Historia de Usuario		
Número: 4	**Usuario:** Cliente	**Prioridad:** Alta
Título: Aumentar objetos tipo Imagen		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada pueda mostrar elementos aumentados de tipo Imagen		
Observaciones: La imagen será descargada a partir de una URL definida en el archivo ARML2.0. La URL de la imagen debe indicarse en el atributo href.		

Historia de Usuario		
Número: 5	**Usuario:** Cliente	**Prioridad:** Alta
Título: Registro de marcadores de realidad aumentada		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada pueda reconocer más de un marcador de realidad aumentada		
Observaciones: Los marcadores deberán estar descritos en el archivo ARML2.0 mediante el atributo URI del elemento Tracker.		

Historia de Usuario		
Número: 6	**Usuario:** Cliente	**Prioridad:** Media
Título: Deshabilitar marcadores		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada ignore todos los marcadores que se encuentren deshabilitados		
Observaciones: Los marcadores deberán estar deshabilitados en el archivo ARML2.0 mediante el atributo *enabled* con valor *false*. Los elementos aumentados asociados a un marcador deshabilitado, también estarán deshabilitados.		

Historia de Usuario		
Número: 7	**Usuario:** Cliente	**Prioridad:** Media
Título: Habilitar marcadores		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada muestre siempre los marcadores de realidad aumentada habilitados		
Observaciones: Los marcadores se encuentran habilitados por defecto. También se pueden habilitar mediante el atributo *enabled* del elemento Anchor con valor *true*.		

Historia de Usuario		
Número: 8	**Usuario:** Cliente	**Prioridad:** Media
Título: Deshabilitar elementos aumentados		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada ignore los elementos aumentados deshabilitados y no los muestre en pantalla cuando se reconozca el marcador.		
Observaciones: Los elementos aumentados se encontrarán deshabilitados mediante el atributo *enabled* del elemento *VisualAsset* con valor *false.*		

Historia de Usuario		
Número: 9	**Usuario:** Cliente	**Prioridad:** Media
Título: Habilitar elementos aumentados		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada reconozca los elementos aumentados que se encuentren habilitados.		
Observaciones: Los elementos aumentados se encontrarán habilitados por defecto. También pude ocurrir la habilitación mediante el atributo *enabled* del elemento *VisualAsset* con valor *true.*		

Historia de Usuario		
Número: 10	**Usuario:** Cliente	**Prioridad:** Media
Título: Reconocer color de texto un elemento Texto		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada reconozca el color de texto que brinde a los elementos de tipo texto.		
Observaciones: El color de texto se especifica mediante el atributo *Style* especificando un valor con el formato: *font-color:#RRGGBB;* El color por defecto para los elementos que no especifican este valor es Negro.		

Historia de Usuario		
Número: 11	**Usuario:** Cliente	**Prioridad:** Media
Título: Reconocer color de fondo de un elemento Texto		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada reconozca el color de fondo que brinde a los elementos de tipo texto.		
Observaciones: El color de texto se especifica mediante el atributo *Style* especificando un valor con el formato: *background-color:#RRGGBB;* El color por defecto para los elementos que no especifican este valor es Transparente.		

Historia de Usuario		
Número: 12	**Usuario:** Cliente	**Prioridad:** Alta
Título: Uso de cámara posterior		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada haga uso de la cámara posterior de un dispositivo móvil para realizar el reconocimiento de imágenes.		
Observaciones:		

Historia de Usuario		
Número: 13	**Usuario:** Cliente	**Prioridad:** Alta
Título: Aumentar objetos tipo HTML		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada pueda aumentar objetos tipo HTML.		
Observaciones: Se reconocerá texto HTML mediante una URL descrita en el atributo href y texto HTML inline especificado en el atributo src.		

Historia de Usuario		
Número: 14	**Usuario:** Cliente	**Prioridad:** Media
Título: Tamaño de objetos HTML aumentados		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada cambie la escala y el tamaño del objeto aumentado de acuerdo a lo indicado en el atributo viewport.		
Observaciones: El valor indicado en viewport tendrá valor por defecto 256. Mayores valores disminuirán la escala del objeto aumentado.		

Historia de Usuario		
Número: 15	**Usuario:** Cliente	**Prioridad:** Media
Título: Orientación de los objetos aumentados.		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada realice una rotación de los elementos aumentados de acuerdo a los valores indicados en el atributo Orientation.		
Observaciones: Para marcadores 2D sólo se tomará en cuenta el atributo tilt. Los valores admitidos variarán desde -180 a 180.		

Historia de Usuario		
Número: 16	**Usuario:** Cliente	**Prioridad:** Media
Título: Orden de despliegue de los elementos aumentados		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada presente los objetos aumentados de acuerdo al atributo zOrder.		
Observaciones: Que los elementos aumentados con mayor valor en zOrder oculten a los elementos que tienen menor valor.		

Historia de Usuario		
Número: 17	**Usuario:** Cliente	**Prioridad:** Alta
Título: Presentación de objetos aumentados de acuerdo al marcador.		
Responsable: Angela Córdova		
Descripción: Como cliente quiero que el navegador de realidad aumentada presente los objetos aumentados correctos de acuerdo al marcador de realidad aumentada que se visualice.		
Observaciones: La relación entre los elementos aumentados y los marcadores se da mediante el atributo TrackableConfig y el atributo assets de un elemento VisualAsset.		

ANEXO IV: PRUEBAS DE VALIDACIÓN DEL ESTÁNDAR

A continuación se presentan las pruebas desarrolladas para la validación de la implementación descriptiva del estándar.

Elemento	ARELement
Descripción	Elemento del cual derivan todas las clases definidas en el estándar ARML2.0
Propiedad	id
Descripción	Identificador único del elemento ARML

Prueba ID	2.3
Descripción	Identificador invalido
Propósito	Validar que la implementación no permite elementos con identificador user
Método	La implementación ignora todos los elementos cuyo id es user

Nro. Prueba:	2.3_1
Nombre:	2.3_1_ARMLElement con id user
Pasos	**Resultado esperado**
1.Leer un archivo ARML con un elemento ARMLElement con id = user	El navegador ignora el elemento y no lo muestra en pantalla.
Archivo de Prueba	

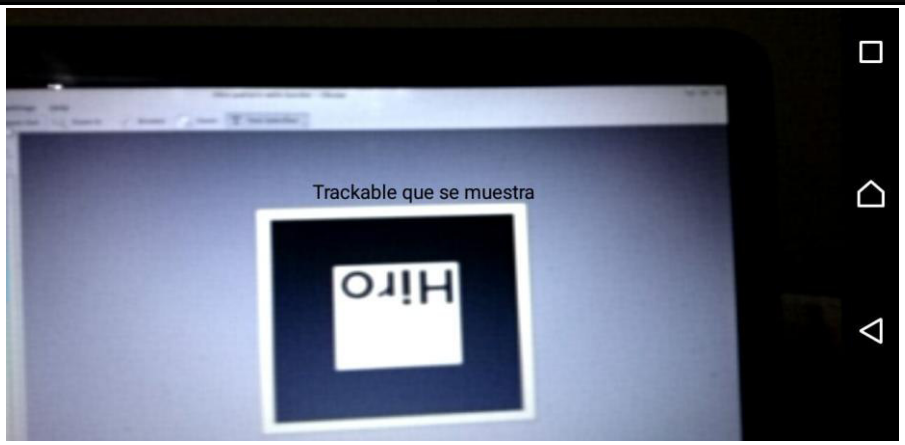
```xml
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements id="user">
    <Trackable>
      <assets>
        <Text id="personalizado">
          <src>Prueba de texto</src>
        </Text>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/emukvb8molhs4wu/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	Conforme

Nro. Prueba:	2.3_2
Nombre:	2.3_2_Trackable con id user
Pasos	**Resultado esperado**
1.Leer un archivo ARML con un elemento Trackable id = user	El navegador ignora el elemento y no lo muestra en pantalla.
Archivo de Prueba	

```xml
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable id="user">
      <assets>
        <Text id="personalizado">
          <src>Trackable ignorado</src>
        </Text>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Trackable id="otroid">
      <assets>
        <Text id="personalizado">
          <src>Trackable que se muestra</src>
        </Text>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/emukvb8molhs4wu/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**

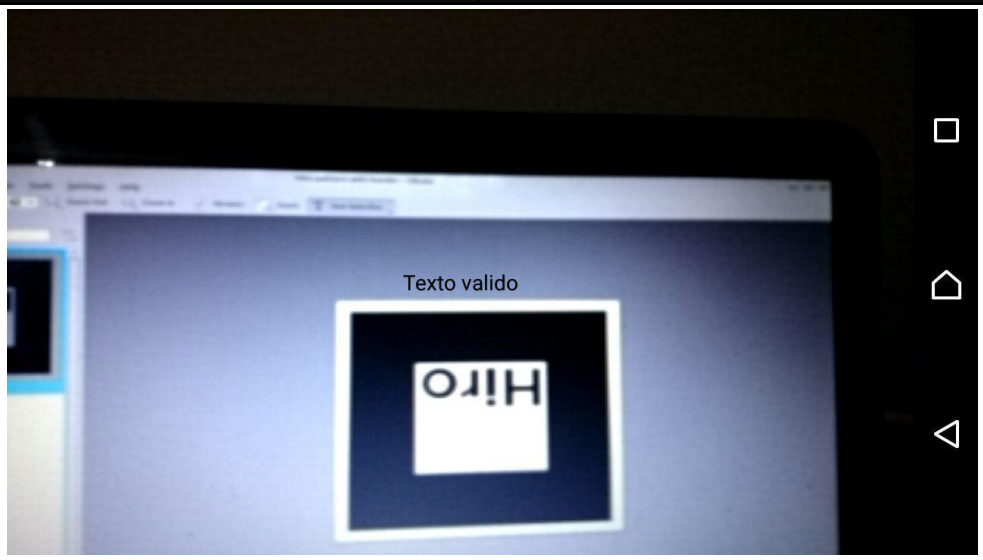


Nro. Prueba:	2.3_3
Nombre:	2.3_3_Texto con id user
Pasos	**Resultado esperado**
1.Leer un archivo ARML que contenga un elemento Texto con id = user	El navegador ignora el elemento y no lo muestra en pantalla.

Archivo de Prueba	

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Text id="user">
          <src>Texto ignorado</src>
        </Text>
        <Text id="nouser">
          <src>Texto valido</src>
        </Text>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/emukvb8molhs4wu/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	Conforme



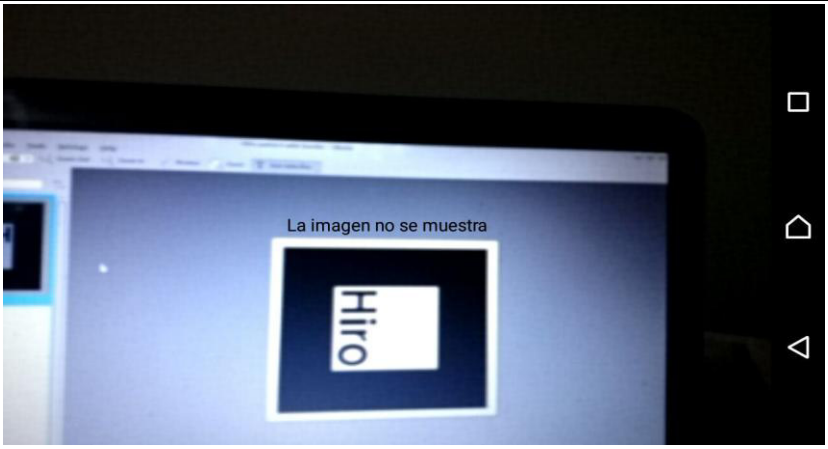
Nro. Prueba:	2.3_4
Nombre:	2.3_4_Label con id user
Pasos	**Resultado esperado**
1.Leer un archivo ARML que contenga un	El navegador ignora el elemento y no lo muestra

elemento Imagen con id = user	en pantalla.

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Image id="user">
          <href xlink:href="https://www.dropbox.com/s/3scpo8is57oxks0/escudo_unmsm.png?dl=1" />
        </Image>
        <Text id="nouser">
          <src>La imagen no se muestra</src>
        </Text>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/emukvb8molhs4wu/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	Conforme

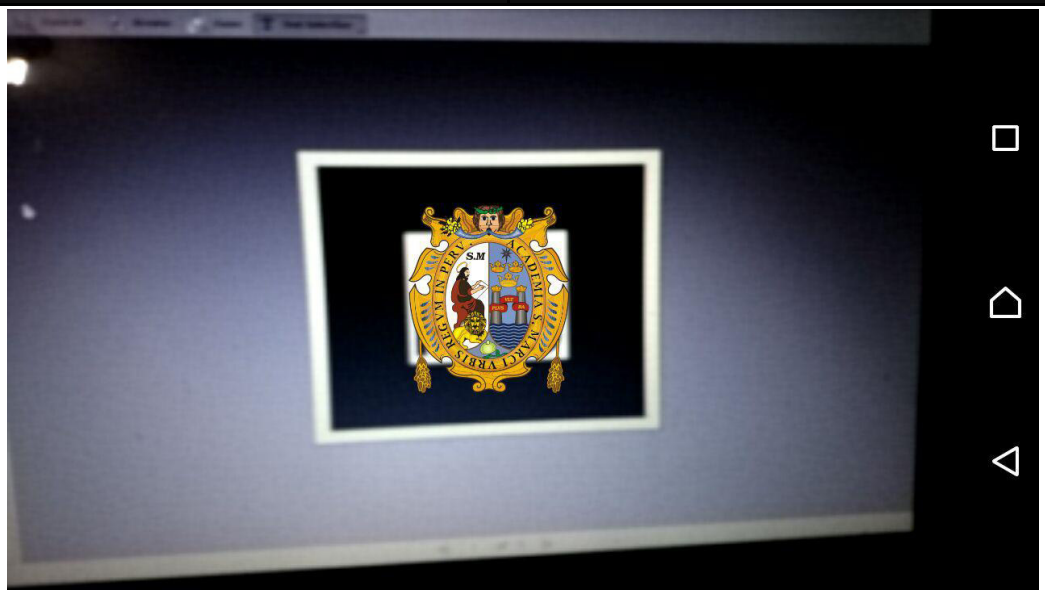


Nro. Prueba:	2.3_5
Nombre:	2.3_5_Label con id user
Pasos	**Resultado esperado**
1.Leer un archivo ARML que contenga un elemento Label con id = user	El navegador ignora el elemento y no lo muestra en pantalla.

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Label id="user">
          <href xlink:href="https://www.google.com" />
        </Label>
        <Image id="imagen">
          <href xlink:href="https://www.dropbox.com/s/3scpo8is57oxks0/escudo_unmsm.png?dl=1" />
        </Image>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/emukvb8molhs4wu/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	Conforme

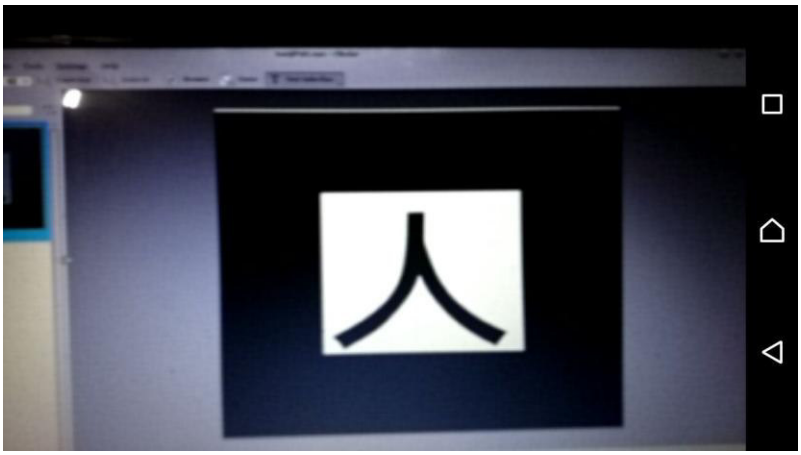


Nro. Prueba:	2.3_6
Nombre:	2.3_6_Tracker con id user
Pasos	**Resultado esperado**
1.Leer un archivo ARML que contenga un elemento Tracker con id = user	El navegador ignora el elemento y no lo muestra en pantalla.

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Label>
          <href xlink:href="https://www.google.com" />
        </Label>
      </assets>
      <config>
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Trackable>
      <assets>
        <Text>
          <src>Este texto no se muestra</src>
        </Text>
      </assets>
      <config>
        <tracker xlink:href="#user" />
        <src>patt.kanji</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/shjk6msgwufm4qs/patt.hiro?dl=1" />
    </Tracker>
    <Tracker id="user">
      <uri xlink:href="https://www.dropbox.com/s/83v18jji50pwzfg/patt.2kanji?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	Conforme



Elemento	Anchor
Descripción	Elemento que representa un objeto del mundo real
Propiedad	enabled
Descripción	Determina si un elemento se encuentra habilitado o no.

Prueba ID	2.5
Descripción	Deshabilitar Anchors
Propósito	Validar que la implementación ignora un Anchor y sus elementos asociados cuando este se encuentra deshabilitado.
Método	La implementación ignora un Anchor y los VisualAssets asociados cuando este tiene el valor false en la propiedad enabled.

Nro. Prueba:	2.5_1
Nombre:	2.5_1_Anchor_deshabilitado
Pasos	**Resultado esperado**
1.Leer un archivo ARML que contenga un elemento Anchor con el elemento enabled false.	Que el Anchor sea ignorado y no se muestren los VisualAssets asociados.

Archivo de Prueba

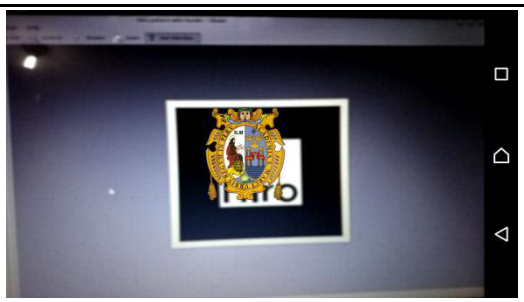
```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <enabled>false</enabled>
      <assets>
        <Text>
          <src>Este Texto no se muestra</src>
        </Text>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>markerID</src>
      </config>
    </Trackable>
    <Trackable>
      <assets>
        <Image>
          <href xlink:href="https://www.dropbox.com/s/3scpo8is57oxks0/escudo_unmsm.png?dl=1" />
        </Image>
      </assets>
      <config>
        <tracker xlink:href="#hiromarker"/>
        <src>markerID</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/cczkqsy0numa9rn/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**

Nro. Prueba:	2.5_2
Nombre:	2.5_2_Anchor_habilitado_por_defecto
Pasos	**Resultado esperado**
1.Leer un archivo ARML que contenga un elemento Anchor que no especifique la propiedad enabled	Que se el navegador interprete como valor por defecto true para el atributo enabled. Que se muestren los VisualAssets habilitados y que se encuentren relacionados en pantalla.
Archivo de Prueba	

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <enabled>false</enabled>
      <assets>
        <Text>
          <src>Este Texto no se muestra</src>
        </Text>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>markerID</src>
      </config>
    </Trackable>
    <Trackable>
      <assets>
        <Image>
          <href xlink:href="https://www.dropbox.com/s/3scpo8is57oxks0/escudo_unmsm.png?dl=1" />
        </Image>
      </assets>
      <config>
        <tracker xlink:href="#hiromarker"/>
        <src>markerID</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/cczkqsy0numa9rn/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**

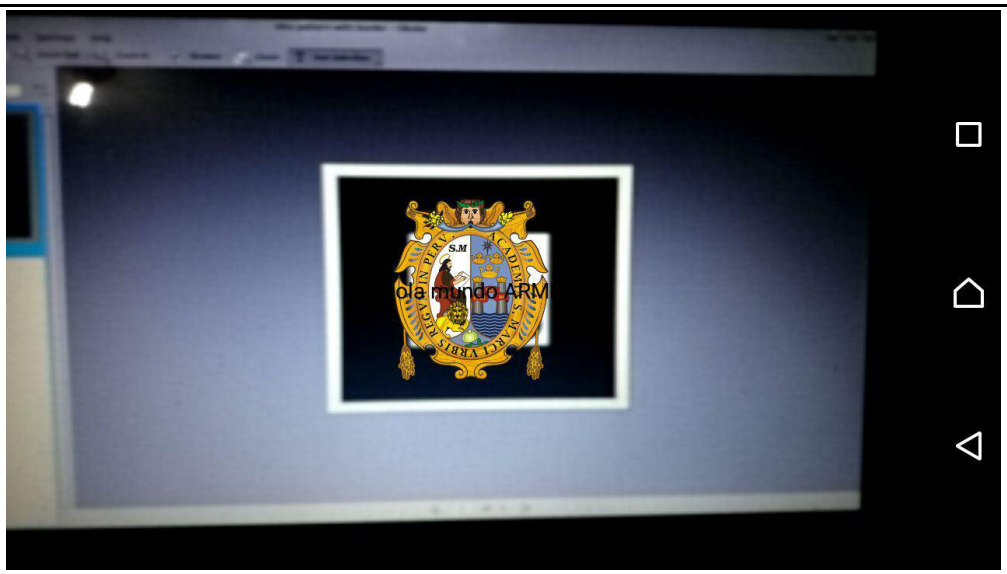


Nro. Prueba:	2.5_3
Nombre:	2.5_3_Anchor_habilitado
Pasos	**Resultado esperado**
1.Leer un archivo ARML que contenga un elemento Anchor con el elemento enabled true	Que los VisualAssets relacionados al Anchor que se encuentren habilitados se muestren en pantalla.

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Image>
          <href xlink:href="https://www.dropbox.com/s/3scpo8is57oxks0/escudo_unmsm.png?dl=1" />
        </Image>
        <Text>
          <src>Hola mundo ARML</src>
        </Text>
      </assets>
      <config>
        <tracker xlink:href="#hiromarker"/>
        <src>markerID</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/cczkqsy0numa9rn/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**



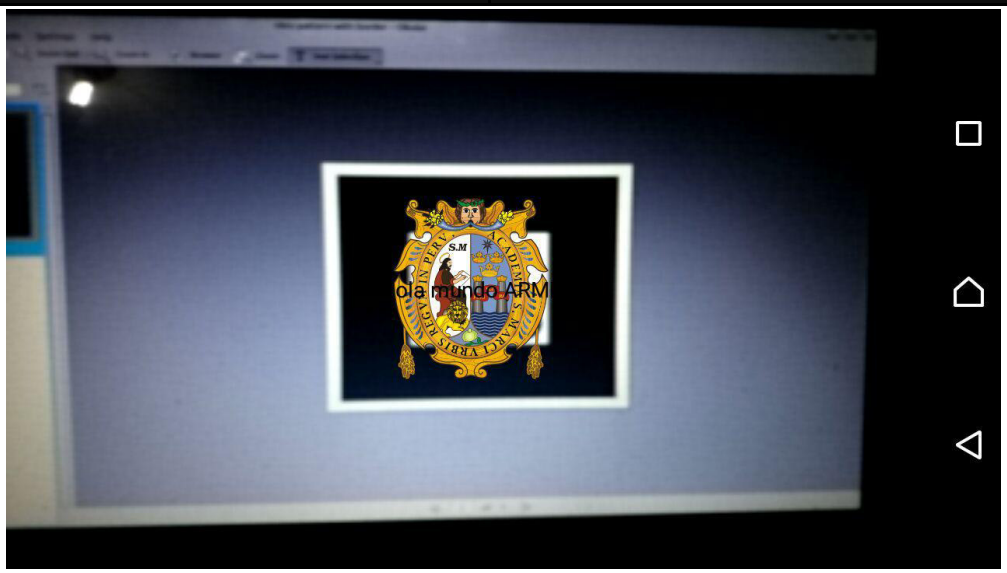
Elemento	Anchor
Descripción	Elemento que representa un objeto del mundo real

Prueba ID	2.6
Descripción	Anchors sin Features
Propósito	Validar que la implementación reconoce Anchors que no están relacionados con un Feature
Método	Verificar que la implementación reconoce Anchors que no están relacionados a Features, incluso si son descendientes directos del elemento ARMLElement

Nro. Prueba:	2.6_1
Nombre:	2.6_1_Anchor_sin_Feature
Pasos	**Resultado esperado**
1.Leer un archivo ARML que contenga un elemento Anchor descendiente directo de un elemento ARMLElement	Que el navegador reconozca correctamente el Anchor y sus elementos relacionados.
Archivo de Prueba	

```xml
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Image>
          <href xlink:href="https://www.dropbox.com/s/3scpo8is57oxks0/escudo_unmsm.png?dl=1" />
        </Image>
        <Text>
          <src>Hola mundo ARML</src>
        </Text>
      </assets>
      <config>
        <tracker xlink:href="#hiromarker"/>
        <src>markerID</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/cczkqsy0numa9rn/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**



Elemento	Tracker
Descripción	Elemento que representa un objeto del mundo real
Propiedad	enabled
Descripción	Determina si un elemento se encuentra habilitado o no.

Prueba ID	2.18
Descripción	Trackers Desconocidos
Propósito	Validar que la implementación no falle en caso exista un Tracker desconocido.
Método	Verificar que la implementación ignora Trackers desconocidos y sus Trackables asociados.

Nro. Prueba:	2.18_1
Nombre:	2.18_1_Marcadores_URL_Host_Desconocidos
Pasos	**Resultado esperado**
1.Leer un archivo ARML que tenga un elemento Tracker con una URL con host desconocido.	Que el navegador ignore el marcador desconocido y sus Trackables asociados

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Text>
          <src>HOLA MUNDO ARML 2.0</src>
        </Text>
      </assets>
      <config > <!-- attribute order optional -->
        <tracker xlink:href="#hiromarker" />
        <src>markerID</src>
      </config>
    </Trackable>
    <Trackable>
      <assets>
        <Text>
          <src>Este texto no se mostrara ya que el marcador es desconocido</src>
        </Text>
      </assets>
      <config>
        <tracker xlink:href="#desconocido" />
        <src>markerID</src>
      </config>
    </Trackable>
    <Tracker id="desconocido">
      <uri xlink:href="https://www.marcadordesconocido.com/patt.hiro?dl=1" />
    </Tracker>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/cczkqsy0numa9rn/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

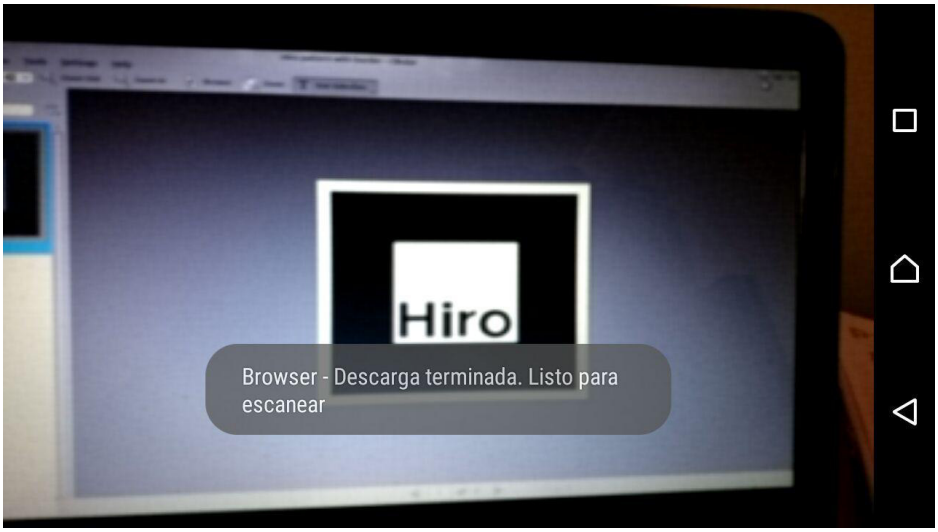
Resultado	**Conforme**

Nro. Prueba:	2.18_2
Nombre:	2.18_2_Marcador_Formato_Desconocido
Pasos	**Resultado esperado**
1.Leer un archivo ARML que tenga un elemento Tracker con un marcador con formato desconocido para el framework ARToolKit	Que el navegador ignore el marcador desconocido y sus Trackables asociados

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Text>
          <src>No se podra mostrar este texto</src>
        </Text>
      </assets>
      <config>
        <tracker xlink:href="#hiromarker" />
        <src>markerID</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/3scpo8is57oxks0/escudo_unmsm.png?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**



Elemento	TrackableConfig
Descripción	Elemento que representa la configuración de un marcador.
Propiedad	order
Descripción	Determina el orden de prioridad de un TrackableConfig

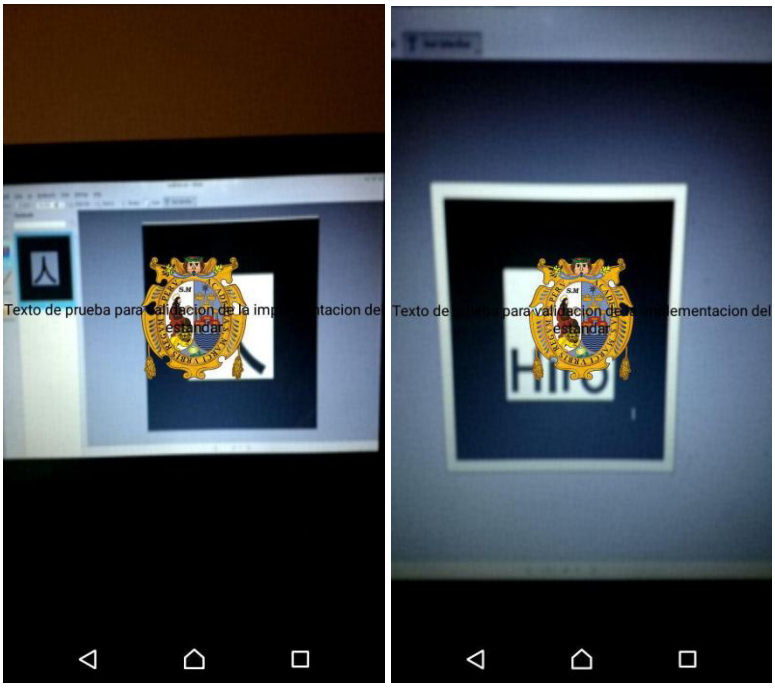
Prueba ID	2.24
Descripción	Orden por defecto de un TrackableConfig
Propósito	Validar que la implementación asigna correctamente el valor de orden por defecto de un TrackableConfig, en caso este no se encuentre asignado
Método	Verificar que la implementación asigna un valor por defecto máximo para el atributo order en caso este no haya sido definido.

Nro. Prueba:	2.24_1
Nombre:	2.24_1_TrackableConfig_order_igual_prioridad
Pasos	**Resultado esperado**

1.Leer un archivo ARML que tenga un elemento Trackable con 2 configuraciones sin el atributo order.	Que el navegador asigne la prioridad máxima a las dos configuraciones. Que el navegador procese todas las configuraciones de la misma prioridad y muestre los visualAssets asociados.
Archivo de Prueba	

```xml
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Image>
          <href xlink:href="https://www.dropbox.com/s/3scpo8is57oxks0/escudo_unmsm.png?dl=1" />
        </Image>
        <Text>
          <src>Texto de prueba para validacion de la implementacion del estandar</src>
        </Text>
      </assets>
      <config>
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
      <config>
        <tracker xlink:href="#kanjimarker" />
        <src>patt.kanji</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/2yqj09q2cwg76op/patt.hiro?dl=1" />
    </Tracker>
    <Tracker id="kanjimarker">
      <uri xlink:href="https://www.dropbox.com/s/83v18jji50pwzfg/patt.2kanji?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**



Elemento	VisualAsset
Descripción	Elemento que representa un objeto aumentado
Propiedad	enabled
Descripción	Determina si un objeto aumentado se encuentra habilitado o no.

Prueba ID	2.30
Descripción	Deshabilitar VisualAssets
Propósito	Validar que la implementación ignora un VisualAsset que se encuentra deshabilitado.
Método	Verificar que la implementación ignora los VisualAssets cuya propiedad enabled tiene el valor false.

Nro. Prueba:	2.30_1
Nombre:	2.30_1 _VisualAsset_deshabilitado

Pasos	Resultado esperado
1.Leer un archivo ARML que tenga un elemento VisualAsset con el valor enabled false.	Que el navegador ignore el VisualAsset y no lo muestre en pantalla.

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
          <assets>
        <Image>
          <enabled>true</enabled>
          <href xlink:href="https://www.dropbox.com/s/3scpo8is57oxks0/escudo_unmsm.png?dl=1" />
        </Image>
        <Text>
          <src>Texto con valor enabled por defecto</src>
        </Text>
        <Text>
          <enabled>false</enabled>
          <src>Este texto no se muestra</src>
        </Text>
      </assets>
      <config > <!-- attribute order optional -->
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/2yqj09q2cwg76op/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	Conforme

Nro. Prueba:	2.30_2
Nombre:	2.30_2 _VisualAsset_habilitado por defecto

Pasos	Resultado esperado
1.Leer un archivo ARML que tenga un elemento VisualAsset sin la propiedad enabled	Que el navegador asigne el valor true a la propiedad enabled y muestre el VisualAsset en la pantalla.

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
          <assets>
        <Image>
          <enabled>true</enabled>
          <href xlink:href="https://www.dropbox.com/s/3scpo8is57oxks0/escudo_unmsm.png?dl=1" />
        </Image>
        <Text>
          <src>Texto con valor enabled por defecto</src>
        </Text>
        <Text>
          <enabled>false</enabled>
          <src>Este texto no se muestra</src>
        </Text>
      </assets>
      <config > <!-- attribute order optional -->
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/2yqj09q2cwg76op/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	Conforme

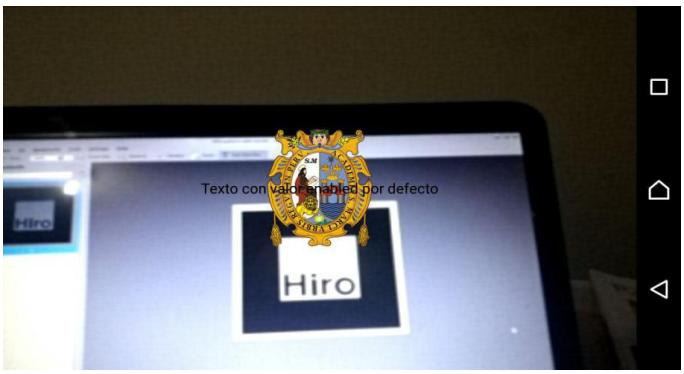
Nro. Prueba:	2.30_2
Nombre:	2.30_2 _VisualAsset_habilitado
Pasos	**Resultado esperado**

1.Leer un archivo ARML que tenga un elemento VisualAsset sin la propiedad enabled	Que el navegador muestre el VisualAsset en la pantalla.

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
          <assets>
        <Image>
          <enabled>true</enabled>
          <href xlink:href="https://www.dropbox.com/s/3scpo8is57oxks0/escudo_unmsm.png?dl=1" />
        </Image>
        <Text>
          <src>Texto con valor enabled por defecto</src>
        </Text>
        <Text>
          <enabled>false</enabled>
          <src>Este texto no se muestra</src>
        </Text>
      </assets>
      <config > <!-- attribute order optional -->
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/2yqj09q2cwg76op/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**



Elemento	VisualAsset
Descripción	Elemento que representa un objeto aumentado
Propiedad	zOrder
Descripción	Determina el orden de despliegue de un VisualAsset

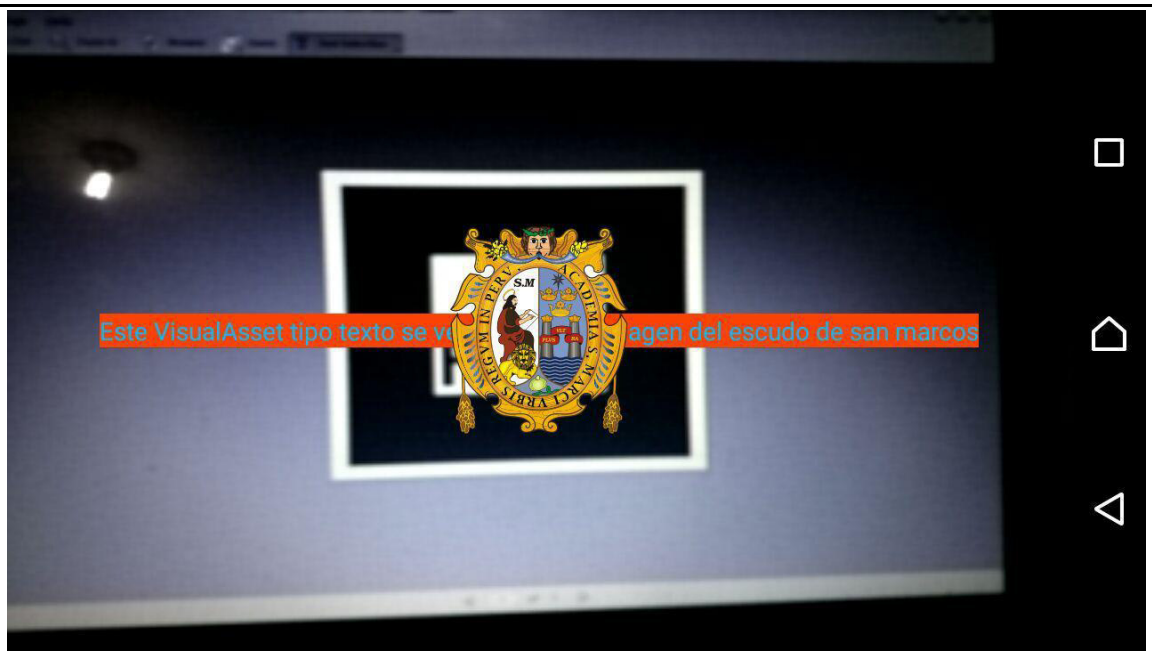
Prueba ID	2.31
Descripción	Orden de proyección
Propósito	Validar que la implementación oculta objetos correctamente.
Método	Verificar que la implementación oculta objetos de acuerdo a la distancia indicada en el valor Zorder

Nro. Prueba:	2.31_1
Nombre:	2.31_1 _Orden_de_proyeccion_VisualAsset
Pasos	**Resultado esperado**
1.Leer un archivo ARML que tenga varios VisualAssets cada uno con diferentes valores en su atributo zOrder	Que el navegador muestre los VisualAssets según el zOrder especificado. Los valor más altos, opacaran a los de menor valor.

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
        <assets>
          <Image>
            <zOrder>3</zOrder>
            <href xlink:href="https://www.dropbox.com/s/3scpo8is57oxks0/escudo_unmsm.png?dl=1" />
          </Image>
          <Text>
            <zOrder>1</zOrder>
            <src>Este VisualAsset tipo texto se ve detras de la imagen del escudo de san marcos</src>
            <style>
                  font-color:#00BDFF;
                  background-color:#FF4200;
            </style>
          </Text>
      </assets>
      <config > <!-- attribute order optional -->
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/2yqj09q2cwg76op/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**



Elemento	Orientation
Descripción	Elemento que permite describir la orientación de un visualAsset
Propiedad	heading
Descripción	Determina el ángulo de rotación de un VisualAsset

Prueba ID	2.56
Descripción	Orientación manual de un VisualAsset.
Propósito	Validar que la ejecución de la orientación manual de VisualAsset.
Método	Verificar que la implementación ejecuta las rotaciones en el orden roll - tilt - heading.

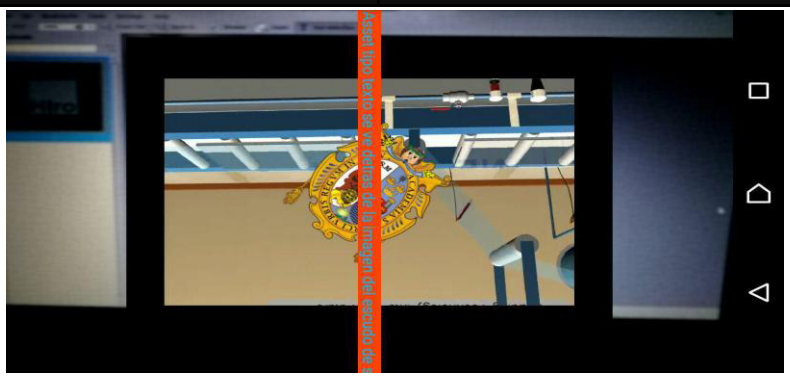
Nro. Prueba:	2.56_1
Nombre:	2.56_1 _Orientacion_VisualAsset
Pasos	**Resultado esperado**

1.Leer un archivo ARML que tenga un VisualAsset y tenga especificado la rotación manual.	Que el navegador muestre los VisualAssets con la rotación correcta en pantalla

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
        <assets>
          <Label>
            <Orientation>
              <heading>180</heading>
            </Orientation>
            <href xlink:href="http://www.pzzr-cas.com/ar/index_video2.html"/>
            <viewportWidth>256</viewportWidth>
          </Label>
          <Image>
            <Orientation>
              <heading>45</heading>
            </Orientation>
            <href xlink:href="https://www.dropbox.com/s/3scpo8is57oxks0/escudo_unmsm.png?dl=1" />
          </Image>
          <Text>
            <Orientation>
              <heading>90</heading>
            </Orientation>
            <src>Este VisualAsset tipo texto se ve detras de la imagen del escudo de san marcos</src>
            <style>
                  font-color:#00BDFF;
                  background-color:#FF4200;
            </style>
          </Text>
      </assets>
      <config > <!-- attribute order optional -->
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/2yqj09q2cwg76op/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**



Elemento	Label
Descripción	Elemento que mostrar objetos de tipo HTML
Propiedad	href
Descripción	Describe una URL para desplegar contenido

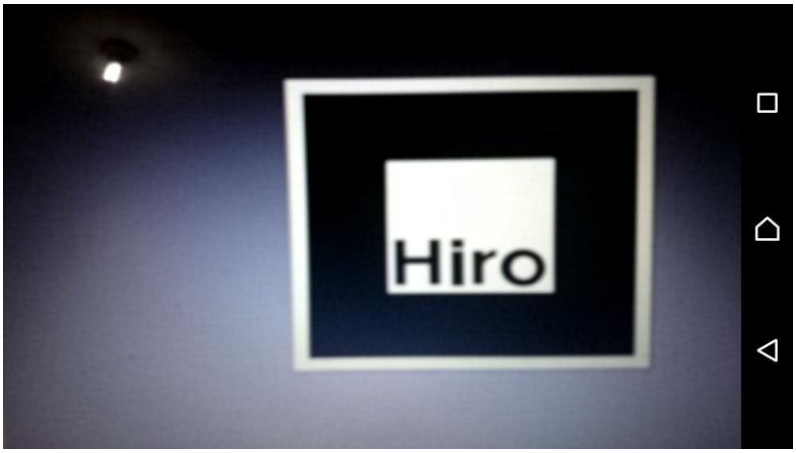
Prueba ID	2.36
Descripción	Contenido de Label requerido
Propósito	Validar que la implementación no falla encaso un Label sea invalido
Método	Verificar que la implementación ignora Labels que no tiene valor en href o en src.

Nro. Prueba:	2.36_1
Nombre:	2.36_1 _Valid_Label
Pasos	**Resultado esperado**
1.Leer un archivo ARML que tenga un elemento Label sin información en los atributos src y href.	Que el navegador ignore el elemento Label.

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <enabled>true</enabled>
      <assets>
        <Label>
          <hyperlinkBehavior>block</hyperlinkBehavior>
          <viewportWidth>256</viewportWidth>
        </Label>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/2yqj09q2cwg76op/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**



Elemento	Label
Descripción	Elemento que mostrar objetos de tipo HTML
Propiedad	src
Descripción	Describe contenido HTML

Prueba ID	2.35
Descripción	Precedencia de atributo src sobre href
Propósito	Validar que la implementación cumple las reglas de precedencia en un Label
Método	Verificar que la implementación brinda mayor precedencia al atributo src sobre el atributo href, en caso ambos tengan valor.

Nro. Prueba:	2.35_1
Nombre:	2.35_1 _Label_Precedencia
Pasos	**Resultado esperado**
1.Leer un archivo ARML que tenga un elemento Label con información en los atributos src y href.	Que el navegador ignore el atributo href y muestre el contenido del atributo Label.
Archivo de Prueba	

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Label>
          <href xlink:href="http://www.pzzr-cas.com/ar/index_video2.html"/>
          <src>
            <![CDATA[
            <html>
              <head>
                <meta charset="uft-8"/>
                <title>Himno de la UNMSM</title>
              </head>
              <body>
                <h1>LETRA DEL HIMNO DE LA DECANA DE AMERICA - UNMSM</h1>
                <h3>Letra: Manuel Tarazona Camacho</h3>
                <h3>Música: Luis Craff Zevallos.</h3>
                <p>
                  Adelante San Marcos glorioso <br/>
                  adelante tú siempre estarás,<br/>
                  porque nadie ha podido vencerte<br/>
                  y jamás nadie te vencerá (bis).<br/>
                  <br/>
                  Es tu nombre un timbre de orgullo<br/>
                  tradición de nobleza y honor.<br/>
                  Siempre grande, siempre limpia<br/>
                  tu bandera más alta estará.<br/>
                  <br/>
                  Sanmarquinos unidos por siempre<br/>
                  con tan grande y profunda misión,<br/>
                  levantemos muy alto la frente<br/>
                  convencidos de nuestro valor.<br/>
                </p>
              </body>
            </html>
            ]]>
          </src>
          <viewportWidth>256</viewportWidth>
        </Label>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/2yqj09q2cwg76op/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**

Prueba ID	2.40
Descripción	Reemplazo de nombre y descripción en Label
Propósito	Validar que la implementación reemplaza correctamente los metadatos de nombre y descripción
Método	Verificar que la implementación reemplace los metadatos $[name] y $[description] con un cadena vacía, en caso el Label no esté relacionado a un Feature

Prueba ID	2.41
Descripción	Reemplazo de propiedad en Label
Propósito	Validar que la implementación reemplaza correctamente los metadatos.
Método	Verificar que la implementación reemplace los metadatos con formato: $[(XPath-Expression)] con un cadena vacía, en caso el Label no esté relacionado a un Feature

Nro. Prueba:	2.40/2.41_1
Nombre:	2.40/41_1 _Label_Metadatos
Pasos	**Resultado esperado**
1.Leer un archivo ARML que tenga un elemento Label que no se encuentre asociado a un elemento Feature y que tenga metadatos $[name] , $[description] y $[(XPath-Expression)].	Que el navegador ignore los metadatos y los reemplace por cadena vacía.
Archivo de Prueba	

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Label>
          <href xlink:href="http://www.pzzr-cas.com/ar/index_video2.html"/>
          <src>
            <![CDATA[
            <html>
              <head>
                <meta charset="uft-8"/>
                <title>Himno de la UNMSM</title>
              </head>
              <body>
                <h1>LETRA DEL HIMNO DE LA DECANA DE AMERICA - UNMSM</h1>
                <h3>Letra: $[autor-letra]</h3>
                <h3>Música: $[name].</h3>
                <p>
                  Adelante $[description] glorioso <br/>
                  adelante tú siempre estarás,<br/>
                  porque nadie ha podido vencerte<br/>
                  y jamás nadie te vencerá (bis).<br/>
                  <br/>
                  Es tu nombre un timbre de orgullo<br/>
                  tradición de nobleza y honor.<br/>
                  Siempre grande, siempre limpia<br/>
                  tu bandera más alta estará.<br/>
                  <br/>
                  Sanmarquinos unidos por siempre<br/>
                  con tan grande y profunda misión,<br/>
                  levantemos muy alto la frente<br/>
                  convencidos de nuestro valor.<br/>
                </p>
              </body>
            </html>
            ]]>
          </src>
          <viewportWidth>256</viewportWidth>
        </Label>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/2yqj09q2cwg76op/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**

Elemento	Label
Descripción	Elemento que permite mostrar objetos de tipo HTML
Propiedad	viewportWidth
Descripción	Elemento que controla el tamaño del contenido del Label y la cantidad de espacio que se encuentra disponible dentro del Label. Si no se establece o se establece en un valor no positivo, se coloca el valor por defecto 256. Cuanto mayor sea el valor, menor será el contenido es mostrado.

Prueba ID	2.39
Descripción	Viewport por defecto en Label
Propósito	Validar que la implementación coloca correctamente el valor por defecto para el atributo viewportWidth de un Label
Método	Verificar que la implementación coloca el valor de 256 cuando la propiedad viewportwidth no tiene valor o tiene un valor negativo.

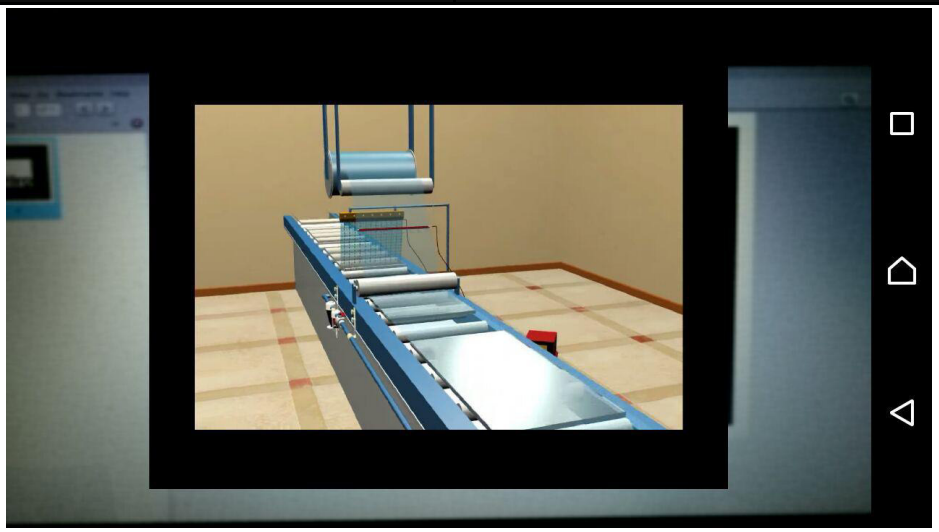
Nro. Prueba:	2.39
Nombre:	2.39_1_Viewport_por_defecto
Pasos	**Resultado esperado**
1.Leer un archivo ARML que tenga un elemento	Que el navegador coloque el valor 256 por

Label que no tenga el atributo viewportWidth definido	defecto.

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Label>
          <href xlink:href="http://www.pzzr-cas.com/ar/index_video2.html"/>
        </Label>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/2yqj09q2cwg76op/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**



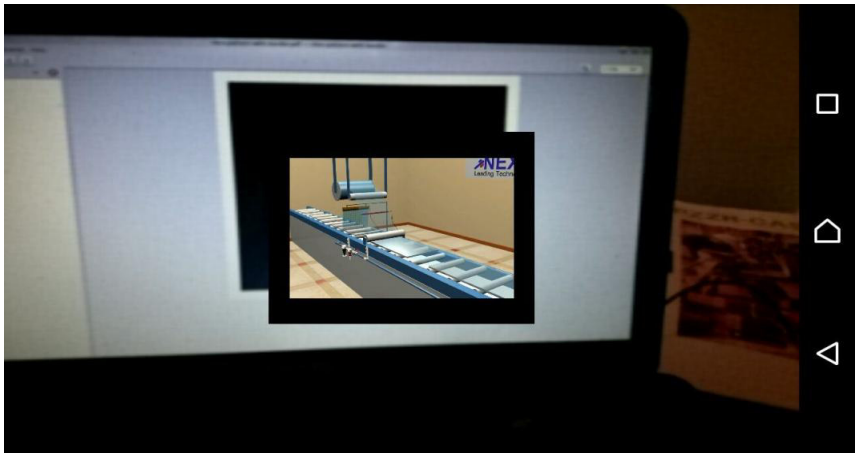
Nro. Prueba:	2.39
Nombre:	2.39_2_Viewport_definido

Pasos	Resultado esperado
1.Leer un archivo ARML que tenga un elemento Label que tenga el atributo viewportWidth definido	Que el navegador coloque el valor definido en el viewport

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Label>
          <href xlink:href="http://www.pzzr-cas.com/ar/index_video2.html"/>
          <viewportWidth>512</viewportWidth>
        </Label>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/2yqj09q2cwg76op/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**



Elemento	Text
Descripción	Elemento que permite mostrar texto plano.
Propiedad	src
Descripción	Indica el contenido del texto

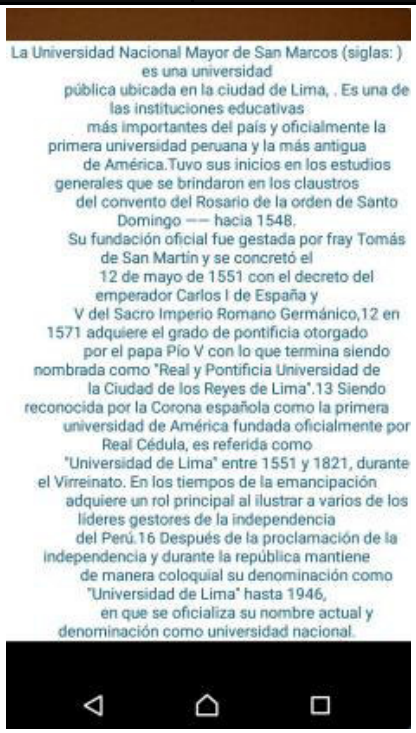
Prueba ID	2.43
Descripción	Reemplazo de los metadatos nombre y descripción
Propósito	Validar que la implementación reemplaza correctamente los metadatos de nombre y descripción
Método	Verificar que la implementación reemplaza cualquier ocurrencia de $[name] y $[description] con cadena vacía en caso el elemento Text no esté relacionado a ningún label.

Prueba ID	2.44
Descripción	Reemplazo de metadatos generales
Propósito	Validar que la implementación reemplaza correctamente los metadatos colocados en el texto
Método	Verificar que la implementación reemplaza cualquier ocurrencia de $[(XPath-Expression)] con cadena vacía en caso el elemento Text no esté relacionado a ningún label.

Nro. Prueba:	2.43/2.44
Nombre:	2.43/2.44_1_ Reemplazo de metadatos
Pasos	**Resultado esperado**
1.Leer un archivo ARML que tenga un elemento Text con un atributo src con los metadatos: $[name], $[description] y $[(XPath-Expression)]	Que el navegador reemplace los metadatos con una cadena vacía.
Archivo de Prueba	

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Text id="personalizado">
          <src>La Universidad Nacional Mayor de San Marcos (siglas: $[name]) es una universidad
             pública ubicada en la ciudad de Lima, $[Pais]. Es una de las instituciones educativas
             más importantes del país y oficialmente la primera universidad peruana y la más antigua
             de América.Tuvo sus inicios en los estudios generales que se brindaron en los claustros
             del convento del Rosario de la orden de Santo Domingo —$[description]— hacia 1548.
             Su fundación oficial fue gestada por fray Tomás de San Martín y se concretó el
             12 de mayo de 1551 con el decreto del emperador Carlos I de España y
             V del Sacro Imperio Romano Germánico,12 en 1571 adquiere el grado de pontificia otorgado
             por el papa Pío V con lo que termina siendo nombrada como "Real y Pontificia Universidad de
             la Ciudad de los Reyes de Lima".13 Siendo reconocida por la Corona española como la primera
             universidad de América fundada oficialmente por Real Cédula, es referida como
             "Universidad de Lima" entre 1551 y 1821, durante el Virreinato. En los tiempos de la emancipación
             adquiere un rol principal al ilustrar a varios de los líderes gestores de la independencia
             del Perú.16 Después de la proclamación de la independencia y durante la república mantiene
             de manera coloquial su denominación como "Universidad de Lima" hasta 1946,
             en que se oficializa su nombre actual y denominación como universidad nacional.</src>
          <style>
                font-color:#175D7B;
                background-color:#FFFFFF;
          </style>
        </Text>
      </assets>
    <config >
      <tracker xlink:href="#hiromarker" />
      <src>patt.hiro</src>
    </config>
  </Trackable>
  <Tracker id="hiromarker">
    <uri xlink:href="https://www.dropbox.com/s/2yqj09q2cwg76op/patt.hiro?dl=1" />
  </Tracker>
 </ARElements>
</arml>
```

Resultado	**Conforme**



Elemento	Text
Descripción	Elemento que permite mostrar texto plano.
Propiedad	style
Descripción	Estilo en línea para el elemento Text. Solo admite dos valores: font-color, background-color.

Prueba ID	2.45
Descripción	Color de fuente por defecto
Propósito	Validar que la implementación coloca el color de fuente por defecto correctamente.
Método	Verificar que la implementación coloca como color por defecto para la fuente, el color negro.

Prueba ID	2.46
Descripción	Color de fondo por defecto
Propósito	Validar que la implementación coloca el color de fondo por defecto correctamente.
Método	Verificar que la implementación coloca como color de fondo por defecto para la fuente el color transparente.

Nro. Prueba:	2.45/2.46
Nombre:	2.45/2.46_1_ Colores por defecto
Pasos	**Resultado esperado**
1.Leer un archivo ARML que tenga un elemento Text y no especifique estilo.	Que el navegador coloque el color de texto y el color de fondo por defecto correctamente.
Archivo de Prueba	

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Text id="personalizado">
          <src>Texto con color de fondo y color de fuente por defecto</src>
        </Text>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/2yqj09q2cwg76op/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**

Elemento	Text
Descripción	Elemento que permite mostrar texto plano.
Propiedad	href
Descripción	Describe la ruta donde se encuentra la imagen

Prueba ID	2.47
Descripción	Imagen con formato invalido
Propósito	Validar que la imagen no falla cuando el formato de la imagen es inválido.
Método	Verificar que la implementación ignora un formato de imagen inválido.

Nro. Prueba:	2.47
Nombre:	2.47_1_ Colores por defecto
Pasos	**Resultado esperado**
1.Leer un archivo ARML que tenga un elemento Imagen no soportado	Que el navegador ignore dicho elemento.

Archivo de Prueba

```
<?xml version="1.0" encoding="utf-8"?>
<arml xmlns="http://www.opengis.net/arml/2.0"
      xmlns:xlink="http://www.w3.org/1999/xlink">
  <ARElements>
    <Trackable>
      <assets>
        <Image>
          <href xlink:href="https://www.dropbox.com/s/3pcdv7entme7p27/err.marker?dl=1" />
        </Image>
        <Text id="personalizado">
          <src>La imagen no se muestra</src>
        </Text>
      </assets>
      <config >
        <tracker xlink:href="#hiromarker" />
        <src>patt.hiro</src>
      </config>
    </Trackable>
    <Tracker id="hiromarker">
      <uri xlink:href="https://www.dropbox.com/s/61m2z2iv4kg3hdf/patt.hiro?dl=1" />
    </Tracker>
  </ARElements>
</arml>
```

Resultado	**Conforme**